

---

**STM32L4Rxxx and STM32L4Sxxx advanced Arm<sup>®</sup>-based  
32-bit MCUs**

---

## Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32L4Rxxx and STM32L4Sxxx microcontrollers memory and peripherals.

The STM32L4Rxxx and STM32L4Sxxx are families of microcontrollers with different memory sizes, packages and peripherals.

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

For information on the Arm<sup>®</sup> Cortex<sup>®</sup>-M4 core, please refer to the Cortex<sup>®</sup>-M4 Technical Reference Manual.

## Related documents

- Cortex<sup>®</sup>-M4 Technical Reference Manual, available from: <http://infocenter.arm.com>
- STM32L4S5xx, STM32L4S7xx and STM32L4S9xx datasheet
- STM32L4R5xx, STM32L4R7xx and STM32L4R9xx datasheet
- STM32F3, STM32F4, STM32L4 and STM32L4+ Series Cortex<sup>®</sup>-M4 (PM0214)

# Contents

<b>1</b>	<b>Documentation conventions</b>	<b>75</b>
1.1	General information	75
1.2	List of abbreviations for registers	75
1.3	Glossary	76
1.4	Availability of peripherals	76
<b>2</b>	<b>System and memory overview</b>	<b>77</b>
2.1	System architecture	77
2.1.1	I-bus	78
2.1.2	D-bus	78
2.1.3	S-bus	78
2.1.4	DMA-bus	79
2.1.5	DMA2D-bus	79
2.1.6	LCD-TFT Controller DMA bus	79
2.1.7	SDMMC Controller DMA bus	79
2.1.8	GFXMMU-bus	79
2.1.9	BusMatrix	79
2.2	Memory organization	81
2.2.1	Introduction	81
2.2.2	Memory map and register boundary addresses	82
2.3	Bit banding	88
2.4	Embedded SRAM	89
2.4.1	SRAM2 parity check	89
2.4.2	SRAM2 Write protection	90
2.4.3	SRAM2 Read protection	92
2.4.4	SRAM2 Erase	92
2.5	Flash memory overview	92
2.6	Boot configuration	93
2.6.1	Boot configuration	93
<b>3</b>	<b>Embedded Flash memory (FLASH)</b>	<b>96</b>
3.1	Introduction	96
3.2	FLASH main features	96



3.3	FLASH functional description	97
3.3.1	Flash memory organization	97
3.3.2	Error code correction (ECC)	101
3.3.3	Read access latency	102
3.3.4	Adaptive real-time memory accelerator (ART Accelerator™)	103
3.3.5	Flash program and erase operations	106
3.3.6	Flash main memory erase sequences	<b>107</b>
3.3.7	Flash main memory programming sequences	108
3.3.8	Read-while-write (RWW) available only in Dual-bank mode (DBANK=1)	112
3.4	FLASH option bytes	114
3.4.1	Option bytes description	114
3.4.2	Option bytes programming	121
3.5	FLASH memory protection	124
3.5.1	Read protection (RDP)	124
3.5.2	Proprietary code readout protection (PCROP)	127
3.5.3	Write protection (WRP)	129
3.6	FLASH interrupts	130
3.7	FLASH registers	131
3.7.1	Flash access control register (FLASH_ACR)	131
3.7.2	Flash Power-down key register (FLASH_PDKEYR)	132
3.7.3	Flash key register (FLASH_KEYR)	133
3.7.4	Flash option key register (FLASH_OPTKEYR)	133
3.7.5	Flash status register (FLASH_SR)	134
3.7.6	Flash control register (FLASH_CR)	136
3.7.7	Flash ECC register (FLASH_ECCR)	138
3.7.8	Flash option register (FLASH_OPTR)	140
3.7.9	Flash PCROP1 Start address register (FLASH_PCROP1SR)	142
3.7.10	Flash PCROP1 End address register (FLASH_PCROP1ER)	142
3.7.11	Flash WRP1 area A address register (FLASH_WRP1AR)	143
3.7.12	Flash WRP1 area B address register (FLASH_WRP1BR)	144
3.7.13	Flash PCROP2 Start address register (FLASH_PCROP2SR)	144
3.7.14	Flash PCROP2 End address register (FLASH_PCROP2ER)	145
3.7.15	Flash WRP2 area A address register (FLASH_WRP2AR)	145
3.7.16	Flash WRP2 area B address register (FLASH_WRP2BR)	146
3.7.17	Flash configuration register (FLASH_CFGR)	147
3.7.18	FLASH register map	148

<b>4</b>	<b>Firewall (FW)</b>	<b>150</b>
4.1	Introduction	150
4.2	Firewall main features	150
4.3	Firewall functional description	151
4.3.1	Firewall AMBA bus snoop	151
4.3.2	Functional requirements	151
4.3.3	Firewall segments	152
4.3.4	Segment accesses and properties	153
4.3.5	Firewall initialization	154
4.3.6	Firewall states	155
4.4	Firewall registers	157
4.4.1	Code segment start address (FW_CSSA)	157
4.4.2	Code segment length (FW_CSL)	157
4.4.3	Non-volatile data segment start address (FW_NVDSOA)	158
4.4.4	Non-volatile data segment length (FW_NVDSL)	158
4.4.5	Volatile data segment start address (FW_VSOA)	159
4.4.6	Volatile data segment length (FW_VDSL)	159
4.4.7	Configuration register (FW_CR)	160
4.4.8	Firewall register map	162
<b>5</b>	<b>Power control (PWR)</b>	<b>163</b>
5.1	Power supplies	163
5.1.1	Independent analog peripherals supply	165
5.1.2	Independent I/O supply rail	166
5.1.3	Independent USB transceivers supply	166
5.1.4	Independent DSI supply	166
5.1.5	Battery backup domain	167
5.1.6	Voltage regulator	168
5.1.7	VDD12 domain	169
5.1.8	Dynamic voltage scaling management	170
5.2	Power supply supervisor	173
5.2.1	Power-on reset (POR) / power-down reset (PDR) / brown-out reset (BOR)	173
5.2.2	Programmable voltage detector (PVD)	173
5.2.3	Peripheral Voltage Monitoring (PVM)	174
5.3	Low-power modes	175

5.3.1	Run mode	182
5.3.2	Low-power run mode (LP run)	182
5.3.3	Low-power modes	183
5.3.4	Sleep mode	184
5.3.5	Low-power sleep mode (LP sleep)	185
5.3.6	Stop 0 mode	186
5.3.7	Stop 1 mode	188
5.3.8	Stop 2 mode	189
5.3.9	Standby mode	191
5.3.10	Shutdown mode	194
5.3.11	Auto-wakeup from low-power mode	195
5.4	PWR registers	196
5.4.1	Power control register 1 (PWR_CR1)	196
5.4.2	Power control register 2 (PWR_CR2)	197
5.4.3	Power control register 3 (PWR_CR3)	198
5.4.4	Power control register 4 (PWR_CR4)	199
5.4.5	Power status register 1 (PWR_SR1)	200
5.4.6	Power status register 2 (PWR_SR2)	201
5.4.7	Power status clear register (PWR_SCR)	203
5.4.8	Power Port A pull-up control register (PWR_PUCRA)	204
5.4.9	Power Port A pull-down control register (PWR_PDCRA)	204
5.4.10	Power Port B pull-up control register (PWR_PUCRB)	205
5.4.11	Power Port B pull-down control register (PWR_PDCRB)	205
5.4.12	Power Port C pull-up control register (PWR_PUCRC)	206
5.4.13	Power Port C pull-down control register (PWR_PDCRC)	206
5.4.14	Power Port D pull-up control register (PWR_PUCRD)	207
5.4.15	Power Port D pull-down control register (PWR_PDCRD)	207
5.4.16	Power Port E pull-up control register (PWR_PUCRE)	208
5.4.17	Power Port E pull-down control register (PWR_PDCRE)	208
5.4.18	Power Port F pull-up control register (PWR_PUCRF)	208
5.4.19	Power Port F pull-down control register (PWR_PDCRF)	209
5.4.20	Power Port G pull-up control register (PWR_PUCRG)	209
5.4.21	Power Port G pull-down control register (PWR_PDCRG)	210
5.4.22	Power Port H pull-up control register (PWR_PUCRH)	210
5.4.23	Power Port H pull-down control register (PWR_PDCRH)	211
5.4.24	Power Port I pull-up control register (PWR_PUCRI)	211
5.4.25	Power Port I pull-down control register (PWR_PDCRI)	212

5.4.26	PWR control register (PWR_CR5) . . . . .	212
5.4.27	PWR register map and reset value table . . . . .	214
<b>6</b>	<b>Reset and clock control (RCC) . . . . .</b>	<b>216</b>
6.1	Reset . . . . .	216
6.1.1	Power reset . . . . .	216
6.1.2	System reset . . . . .	216
6.1.3	Backup domain reset . . . . .	217
6.2	Clocks . . . . .	218
6.2.1	HSE clock . . . . .	222
6.2.2	HSI16 clock . . . . .	224
6.2.3	MSI clock . . . . .	225
6.2.4	HSI48 clock . . . . .	225
6.2.5	PLL . . . . .	226
6.2.6	LSE clock . . . . .	227
6.2.7	LSI clock . . . . .	227
6.2.8	System clock (SYSCLK) selection . . . . .	227
6.2.9	Clock source frequency versus voltage scaling . . . . .	228
6.2.10	Clock security system (CSS) . . . . .	228
6.2.11	Clock security system on LSE . . . . .	229
6.2.12	ADC clock . . . . .	229
6.2.13	RTC clock . . . . .	229
6.2.14	Timer clock . . . . .	230
6.2.15	Watchdog clock . . . . .	230
6.2.16	Clock-out capability . . . . .	231
6.2.17	Internal/external clock measurement with TIM15/TIM16/TIM17 . . . . .	231
6.2.18	Peripheral clock enable register (RCC_AHBxENR, RCC_APBxENRy) . . . . .	234
6.3	Low-power modes . . . . .	234
6.4	RCC registers . . . . .	235
6.4.1	Clock control register (RCC_CR) . . . . .	235
6.4.2	Internal clock sources calibration register (RCC_ICSCR) . . . . .	238
6.4.3	Clock configuration register (RCC_CFGR) . . . . .	238
6.4.4	PLL configuration register (RCC_PLLCFGR) . . . . .	241
6.4.5	PLLSAI1 configuration register (RCC_PLLSAI1CFGR) . . . . .	243
6.4.6	PLLSAI2 configuration register (RCC_PLLSAI2CFGR) . . . . .	247
6.4.7	Clock interrupt enable register (RCC_CIER) . . . . .	249

6.4.8	Clock interrupt flag register (RCC_CIFR) .....	251
6.4.9	Clock interrupt clear register (RCC_CICR) .....	252
6.4.10	AHB1 peripheral reset register (RCC_AHB1RSTR) .....	254
6.4.11	AHB2 peripheral reset register (RCC_AHB2RSTR) .....	255
6.4.12	AHB3 peripheral reset register (RCC_AHB3RSTR) .....	257
6.4.13	APB1 peripheral reset register 1 (RCC_APB1RSTR1) .....	258
6.4.14	APB1 peripheral reset register 2 (RCC_APB1RSTR2) .....	260
6.4.15	APB2 peripheral reset register (RCC_APB2RSTR) .....	261
6.4.16	AHB1 peripheral clock enable register (RCC_AHB1ENR) .....	263
6.4.17	AHB2 peripheral clock enable register (RCC_AHB2ENR) .....	264
6.4.18	AHB3 peripheral clock enable register (RCC_AHB3ENR) .....	266
6.4.19	APB1 peripheral clock enable register 1 (RCC_APB1ENR1) .....	267
6.4.20	APB1 peripheral clock enable register 2 (RCC_APB1ENR2) .....	269
6.4.21	APB2 peripheral clock enable register (RCC_APB2ENR) .....	271
6.4.22	AHB1 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB1SMENR) .....	272
6.4.23	AHB2 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB2SMENR) .....	274
6.4.24	AHB3 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB3SMENR) .....	276
6.4.25	APB1 peripheral clocks enable in Sleep and Stop modes register 1 (RCC_APB1SMENR1) .....	277
6.4.26	APB1 peripheral clocks enable in Sleep and Stop modes register 2 (RCC_APB1SMENR2) .....	279
6.4.27	APB2 peripheral clocks enable in Sleep and Stop modes register (RCC_APB2SMENR) .....	281
6.4.28	Peripherals independent clock configuration register (RCC_CCIPR) .	282
6.4.29	Backup domain control register (RCC_BDCR) .....	285
6.4.30	Control/status register (RCC_CSR) .....	286
6.4.31	Clock recovery RC register (RCC_CRRCR) .....	288
6.4.32	Peripherals independent clock configuration register (RCC_CCIPR2)	289
6.4.33	RCC register map .....	291
<b>7</b>	<b>Clock recovery system (CRS) .....</b>	<b>296</b>
7.1	Introduction .....	296
7.2	CRS main features .....	296
7.3	CRS functional description .....	297
7.3.1	CRS block diagram .....	297
7.3.2	Synchronization input .....	297

- 7.3.3 Frequency error measurement ..... 298
- 7.3.4 Frequency error evaluation and automatic trimming ..... 299
- 7.3.5 CRS initialization and configuration ..... 299
- 7.4 CRS low-power modes ..... 300
- 7.5 CRS interrupts ..... 300
- 7.6 CRS registers ..... 301
  - 7.6.1 CRS control register (CRS\_CR) ..... 301
  - 7.6.2 CRS configuration register (CRS\_CFGR) ..... 302
  - 7.6.3 CRS interrupt and status register (CRS\_ISR) ..... 303
  - 7.6.4 CRS interrupt flag clear register (CRS\_ICR) ..... 305
  - 7.6.5 CRS register map ..... 306
  
- 8 General-purpose I/Os (GPIO) ..... 307**
  - 8.1 Introduction ..... 307
  - 8.2 GPIO main features ..... 307
  - 8.3 GPIO functional description ..... 307
    - 8.3.1 General-purpose I/O (GPIO) ..... 310
    - 8.3.2 I/O pin alternate function multiplexer and mapping ..... 310
    - 8.3.3 I/O port control registers ..... 311
    - 8.3.4 I/O port data registers ..... 311
    - 8.3.5 I/O data bitwise handling ..... 311
    - 8.3.6 GPIO locking mechanism ..... 312
    - 8.3.7 I/O alternate function input/output ..... 312
    - 8.3.8 External interrupt/wakeup lines ..... 312
    - 8.3.9 Input configuration ..... 312
    - 8.3.10 Output configuration ..... 313
    - 8.3.11 Alternate function configuration ..... 314
    - 8.3.12 Analog configuration ..... 315
    - 8.3.13 Using the HSE or LSE oscillator pins as GPIOs ..... 315
    - 8.3.14 Using the GPIO pins in the RTC supply domain ..... 315
    - 8.3.15 Using PH3 as GPIO ..... 316
  - 8.4 GPIO registers ..... 316
    - 8.4.1 GPIO port mode register (GPIOx\_MODER) (x = A to I) ..... 316
    - 8.4.2 GPIO port output type register (GPIOx\_OTYPER) (x = A to I) ..... 317
    - 8.4.3 GPIO port output speed register (GPIOx\_OSPEEDR) (x = A to I) ..... 317

8.4.4	GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to I) . . . . .	317
8.4.5	GPIO port input data register (GPIOx_IDR) (x = A to I) . . . . .	318
8.4.6	GPIO port output data register (GPIOx_ODR) (x = A to I) . . . . .	318
8.4.7	GPIO port bit set/reset register (GPIOx_BSRR) (x = A to I) . . . . .	319
8.4.8	GPIO port configuration lock register (GPIOx_LCKR) (x = A to I) . . . . .	319
8.4.9	GPIO alternate function low register (GPIOx_AFLR) (x = A to I) . . . . .	320
8.4.10	GPIO alternate function high register (GPIOx_AFRH) (x = A to I) . . . . .	321
8.4.11	GPIO port bit reset register (GPIOx_BRR) (x = A to I) . . . . .	322
8.4.12	GPIO register map . . . . .	323
<b>9</b>	<b>System configuration controller (SYSCFG) . . . . .</b>	<b>325</b>
9.1	SYSCFG main features . . . . .	325
9.2	SYSCFG registers . . . . .	325
9.2.1	SYSCFG memory remap register (SYSCFG_MEMRMP) . . . . .	325
9.2.2	SYSCFG configuration register 1 (SYSCFG_CFGR1) . . . . .	326
9.2.3	SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1) . . . . .	328
9.2.4	SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2) . . . . .	330
9.2.5	SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3) . . . . .	331
9.2.6	SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4) . . . . .	333
9.2.7	SYSCFG SRAM2 control and status register (SYSCFG_SCSR) . . . . .	334
9.2.8	SYSCFG configuration register 2 (SYSCFG_CFGR2) . . . . .	335
9.2.9	SYSCFG SRAM2 write protection register (SYSCFG_SWPR) . . . . .	336
9.2.10	SYSCFG SRAM2 key register (SYSCFG_SKR) . . . . .	336
9.2.11	SYSCFG SRAM2 write protection register 2 (SYSCFG_SWPR2) . . . . .	337
9.2.12	SYSCFG register map . . . . .	338
<b>10</b>	<b>Peripherals interconnect matrix . . . . .</b>	<b>340</b>
10.1	Introduction . . . . .	340
10.2	Connection summary . . . . .	340
10.3	Interconnection details . . . . .	341

10.3.1	From timer (TIM1/TIM2/TIM3/TIM4/TIM5/TIM8/TIM15/TIM16/TIM17) to timer (TIM1/TIM2/TIM3/TIM4/TIM5/TIM8/TIM15) . . . . .	341
10.3.2	From timer (TIM1/TIM2/TIM3/TIM4/TIM6/TIM8/TIM15) and EXTI to ADC (ADC1) . . . . .	342
10.3.3	From ADC to timer (TIM1/TIM8) . . . . .	342
10.3.4	From timer (TIM2/TIM4/TIM5/TIM6/TIM7/TIM8) and EXTI to DAC (DAC1/DAC2) . . . . .	343
10.3.5	From timer (TIM1/TIM3/TIM4/TIM6/TIM7/TIM8/TIM16/LPTIM1/LPTIM2) and EXTI to DFSDM1 . . . . .	343
10.3.6	From DFSDM1 to timer (TIM1/TIM8/TIM15/TIM16/TIM17) . . . . .	344
10.3.7	From HSE, LSE, LSI, MSI, MCO, RTC to timer (TIM2/TIM15/TIM16/TIM17) . . . . .	344
10.3.8	From RTC, COMP1, COMP2 to low-power timer (LPTIM1/LPTIM2) . . . . .	344
10.3.9	From timer (TIM1/TIM2/TIM3/TIM8/TIM15) to comparators (COMP1/COMP2) . . . . .	345
10.3.10	From USB to timer (TIM2) . . . . .	345
10.3.11	From internal analog source to ADC and OPAMP (OPAMP1/OPAMP2) . . . . .	346
10.3.12	From comparators (COMP1/COMP2) to timers (TIM1/TIM2/TIM3/TIM8/TIM15/TIM16/TIM17) . . . . .	346
10.3.13	From system errors to timers (TIM1/TIM8/TIM15/TIM16/TIM17) . . . . .	347
10.3.14	From timers (TIM16/TIM17) to IRTIM . . . . .	347
10.3.15	From ADC (ADC1) to DFSDM . . . . .	347
<b>11</b>	<b>Direct memory access controller (DMA) . . . . .</b>	<b>348</b>
11.1	Introduction . . . . .	348
11.2	DMA main features . . . . .	348
11.3	DMA implementation . . . . .	349
11.3.1	DMA1 and DMA2 . . . . .	349
11.3.2	DMA request mapping . . . . .	349
11.4	DMA functional description . . . . .	349
11.4.1	DMA block diagram . . . . .	349
11.4.2	DMA pins and internal signals . . . . .	351
11.4.3	DMA transfers . . . . .	351
11.4.4	DMA arbitration . . . . .	352
11.4.5	DMA channels . . . . .	352
11.4.6	DMA data width, alignment and endianness . . . . .	356
11.4.7	DMA error management . . . . .	357
11.5	DMA interrupts . . . . .	358



11.6	DMA registers	358
11.6.1	DMA interrupt status register (DMA_ISR)	358
11.6.2	DMA interrupt flag clear register (DMA_IFCR)	361
11.6.3	DMA channel x configuration register (DMA_CCRx)	362
11.6.4	DMA channel x number of data to transfer register (DMA_CNDTRx)	365
11.6.5	DMA channel x peripheral address register (DMA_CPARx)	365
11.6.6	DMA channel x memory address register (DMA_CMARx)	366
11.6.7	DMA register map and reset values	366
<b>12</b>	<b>DMA request multiplexer (DMAMUX)</b>	<b>369</b>
12.1	Introduction	369
12.2	DMAMUX main features	369
12.3	DMAMUX implementation	370
12.3.1	DMAMUX instantiation	370
12.3.2	DMAMUX mapping	370
12.4	DMAMUX functional description	373
12.4.1	DMAMUX block diagram	373
12.4.2	DMAMUX signals	374
12.4.3	DMAMUX channels	374
12.4.4	DMAMUX request line multiplexer	374
12.4.5	DMAMUX request generator	377
12.5	DMAMUX interrupts	378
12.6	DMAMUX registers	379
12.6.1	DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR)	379
12.6.2	DMAMUX request line multiplexer interrupt channel status register (DMAMUX_CSR)	380
12.6.3	DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR)	380
12.6.4	DMAMUX request generator channel x configuration register (DMAMUX_RGxCR)	381
12.6.5	DMAMUX request generator interrupt status register (DMAMUX_RGSR)	382
12.6.6	DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR)	382
12.6.7	DMAMUX register map	383
<b>13</b>	<b>Chrom-ART Accelerator™ controller (DMA2D)</b>	<b>385</b>

13.1	DMA2D introduction	385
13.2	DMA2D main features	385
13.3	DMA2D functional description	386
13.3.1	General description	386
13.3.2	DMA2D control	387
13.3.3	DMA2D foreground and background FIFOs	387
13.3.4	DMA2D foreground and background pixel format converter (PFC)	388
13.3.5	DMA2D foreground and background CLUT interface	390
13.3.6	DMA2D blender	391
13.3.7	DMA2D output PFC	392
13.3.8	DMA2D output FIFO	392
13.3.9	DMA2D output FIFO byte reordering	393
13.3.10	DMA2D AHB master port timer	394
13.3.11	DMA2D transactions	395
13.3.12	DMA2D configuration	395
13.3.13	DMA2D transfer control (start, suspend, abort and completion)	399
13.3.14	Watermark	399
13.3.15	Error management	399
13.3.16	AHB dead time	399
13.4	DMA2D interrupts	400
13.5	DMA2D registers	401
13.5.1	DMA2D control register (DMA2D_CR)	401
13.5.2	DMA2D Interrupt Status Register (DMA2D_ISR)	403
13.5.3	DMA2D interrupt flag clear register (DMA2D_IFCR)	404
13.5.4	DMA2D foreground memory address register (DMA2D_FGMAR)	405
13.5.5	DMA2D foreground offset register (DMA2D_FGOR)	405
13.5.6	DMA2D background memory address register (DMA2D_BGMAR)	405
13.5.7	DMA2D background offset register (DMA2D_BGOR)	406
13.5.8	DMA2D foreground PFC control register (DMA2D_FGPFCCR)	407
13.5.9	DMA2D foreground color register (DMA2D_FGCOLOR)	409
13.5.10	DMA2D background PFC control register (DMA2D_BGPFCCR)	410
13.5.11	DMA2D background color register (DMA2D_BGCOLOR)	412
13.5.12	DMA2D foreground CLUT memory address register (DMA2D_FGCMAR)	412
13.5.13	DMA2D background CLUT memory address register (DMA2D_BGCMAR)	413
13.5.14	DMA2D output PFC control register (DMA2D_OPFCCR)	413

	13.5.15 DMA2D output color register (DMA2D_OCOLR) . . . . .	414
	13.5.16 DMA2D output memory address register (DMA2D_OMAR) . . . . .	416
	13.5.17 DMA2D output offset register (DMA2D_OOR) . . . . .	417
	13.5.18 DMA2D number of line register (DMA2D_NLR) . . . . .	417
	13.5.19 DMA2D line watermark register (DMA2D_LWR) . . . . .	418
	13.5.20 DMA2D AHB master timer configuration register (DMA2D_AMTCR) . . . . .	418
	13.5.21 DMA2D register map . . . . .	419
<b>14</b>	<b>Chrom-GRC™ (GFXMMU) . . . . .</b>	<b>421</b>
	14.1 Introduction . . . . .	421
	14.2 Chrom-GRC™ main features . . . . .	421
	14.3 Chrom-GRC™ functional and architectural description . . . . .	422
	14.3.1 Virtual memory . . . . .	422
	14.3.2 MMU architecture . . . . .	424
	14.4 Graphic MMU interrupts . . . . .	428
	14.5 Graphic MMU registers . . . . .	429
	14.5.1 Graphic MMU configuration register (GFXMMU_CR) . . . . .	429
	14.5.2 Graphic MMU status register (GFXMMU_SR) . . . . .	430
	14.5.3 Graphic MMU flag clear register (GFXMMU_FCR) . . . . .	430
	14.5.4 Graphic MMU default value register (GFXMMU_DVR) . . . . .	431
	14.5.5 Graphic MMU buffer 0 configuration register (GFXMMU_B0CR) . . . . .	431
	14.5.6 Graphic MMU buffer 1 configuration register (GFXMMU_B1CR) . . . . .	432
	14.5.7 Graphic MMU buffer 2 configuration register (GFXMMU_B2CR) . . . . .	432
	14.5.8 Graphic MMU buffer 3 configuration register (GFXMMU_B3CR) . . . . .	433
	14.5.9 Graphic MMU LUT entry x low (GFXMMU_LUTxL) . . . . .	433
	14.5.10 Graphic MMU LUT entry x high (GFXMMU_LUTxH) . . . . .	433
	14.5.11 Graphic MMU register map . . . . .	435
<b>15</b>	<b>Nested vectored interrupt controller (NVIC) . . . . .</b>	<b>436</b>
	15.1 NVIC main features . . . . .	436
	15.2 SysTick calibration value register . . . . .	436
	15.3 Interrupt and exception vectors . . . . .	437
<b>16</b>	<b>Extended interrupts and events controller (EXTI) . . . . .</b>	<b>441</b>
	16.1 Introduction . . . . .	441
	16.2 EXTI main features . . . . .	441

16.3	EXTI functional description	441
16.3.1	EXTI block diagram	442
16.3.2	Wakeup event management	442
16.3.3	Peripherals asynchronous Interrupts	443
16.3.4	Hardware interrupt selection	443
16.3.5	Hardware event selection	443
16.3.6	Software interrupt/event selection	443
16.4	EXTI interrupt/event line mapping	443
16.5	EXTI registers	446
16.5.1	Interrupt mask register 1 (EXTI_IMR1)	446
16.5.2	Event mask register 1 (EXTI_EMR1)	446
16.5.3	Rising trigger selection register 1 (EXTI_RTZR1)	446
16.5.4	Falling trigger selection register 1 (EXTI_FTZR1)	447
16.5.5	Software interrupt event register 1 (EXTI_SWIER1)	448
16.5.6	Pending register 1 (EXTI_PR1)	449
16.5.7	Interrupt mask register 2 (EXTI_IMR2)	449
16.5.8	Event mask register 2 (EXTI_EMR2)	450
16.5.9	Rising trigger selection register 2 (EXTI_RTZR2)	450
16.5.10	Falling trigger selection register 2 (EXTI_FTZR2)	451
16.5.11	Software interrupt event register 2 (EXTI_SWIER2)	451
16.5.12	Pending register 2 (EXTI_PR2)	452
16.5.13	EXTI register map	453
<b>17</b>	<b>Cyclic redundancy check calculation unit (CRC)</b>	<b>454</b>
17.1	Introduction	454
17.2	CRC main features	454
17.3	CRC functional description	455
17.3.1	CRC block diagram	455
17.3.2	CRC internal signals	455
17.3.3	CRC operation	455
17.4	CRC registers	457
17.4.1	Data register (CRC_DR)	457
17.4.2	Independent data register (CRC_IDR)	457
17.4.3	Control register (CRC_CR)	458
17.4.4	Initial CRC value (CRC_INIT)	458
17.4.5	CRC polynomial (CRC_POL)	459

	17.4.6	CRC register map	459
<b>18</b>		<b>Flexible static memory controller (FSMC)</b>	<b>460</b>
	18.1	FSMC main features	460
	18.2	FMC block diagram	461
	18.3	AHB interface	462
	18.3.1	Supported memories and transactions	462
	18.4	External device address mapping	463
	18.4.1	NOR/PSRAM address mapping	464
	18.4.2	NAND Flash memory address mapping	465
	18.5	NOR Flash/PSRAM controller	466
	18.5.1	External memory interface signals	467
	18.5.2	Supported memories and transactions	469
	18.5.3	General timing rules	470
	18.5.4	NOR Flash/PSRAM controller asynchronous transactions	471
	18.5.5	Synchronous transactions	489
	18.5.6	NOR/PSRAM controller registers	496
	18.6	NAND Flash controller	503
	18.6.1	External memory interface signals	504
	18.6.2	NAND Flash supported memories and transactions	506
	18.6.3	Timing diagrams for NAND Flash memory	506
	18.6.4	NAND Flash operations	507
	18.6.5	NAND Flash prewait functionality	508
	18.6.6	Computation of the error correction code (ECC) in NAND Flash memory	509
	18.6.7	NAND Flash controller registers	510
	18.7	FSMC register map	516
<b>19</b>		<b>Octo-SPI interface (OCTOSPI)</b>	<b>518</b>
	19.1	OCTOSPI main features	518
	19.2	OCTOSPI functional description	519
	19.2.1	OCTOSPI regular-command mode	520
	19.2.2	OCTOSPI regular-command mode signal interface	524
	19.2.3	HyperBus™ mode	527
	19.2.4	Common functionality between the regular-command mode and HyperBus™	531
	19.3	OCTOSPI operating modes	533

19.3.1	OCTOSPI indirect mode	533
19.3.2	OCTOSPI status flag polling mode	535
19.3.3	OCTOSPI memory-mapped mode	535
19.4	OCTOSPI configuration	537
19.4.1	OCTOSPI system configuration	537
19.4.2	OCTOSPI device configuration	537
19.4.3	OCTOSPI regular-command mode configuration	538
19.4.4	OCTOSPI HyperBus™ mode configuration	540
19.4.5	OCTOSPI error management	541
19.4.6	OCTOSPI busy bit and abort functionality	541
19.4.7	OCTOSPI reconfiguration or deactivation	542
19.4.8	nCS behavior	542
19.5	OCTOSPI Interrupts	544
19.6	OCTOSPI registers	545
19.6.1	OCTOSPI control register (OCTOSPI_CR)	545
19.6.2	OCTOSPI device configuration register 1 (OCTOSPI_DCR1)	547
19.6.3	OCTOSPI device configuration register 2 (OCTOSPI_DCR2)	549
19.6.4	OCTOSPI device configuration register 3 (OCTOSPI_DCR3)	549
19.6.5	OCTOSPI status register (OCTOSPI_SR)	550
19.6.6	OCTOSPI flag clear register (OCTOSPI_FCR)	551
19.6.7	OCTOSPI data length register (OCTOSPI_DLR)	552
19.6.8	OCTOSPI address register (OCTOSPI_AR)	552
19.6.9	OCTOSPI data register (OCTOSPI_DR)	553
19.6.10	OCTOSPI polling status mask register (OCTOSPI_PSMKR)	553
19.6.11	OCTOSPI polling status match register (OCTOSPI_PSMAR)	554
19.6.12	OCTOSPI polling interval register (OCTOSPI_PIR)	554
19.6.13	OCTOSPI communication configuration register (OCTOSPI_CCR)	554
19.6.14	OCTOSPI timing configuration register (OCTOSPI_TCR)	557
19.6.15	OCTOSPI instruction register (OCTOSPI_IR)	558
19.6.16	OCTOSPI alternate bytes register (OCTOSPI_ABR)	558
19.6.17	OCTOSPI low-power timeout register (OCTOSPI_LPTR)	559
19.6.18	OCTOSPI write communication configuration register (OCTOSPI_WCCR)	559
19.6.19	OCTOSPI write timing configuration register (OCTOSPI_WTCR)	562
19.6.20	OCTOSPI write instruction register (OCTOSPI_WIR)	562
19.6.21	OCTOSPI write alternate bytes register (OCTOSPI_WABR)	563

19.6.22	OCTOSPI HyperBus™ latency configuration register (OCTOSPI_HLCR) .....	563
19.6.23	OCTOSPI register map .....	565
<b>20</b>	<b>OctoSPI IO Manager (OCTOSPIM) .....</b>	<b>568</b>
20.1	Introduction .....	568
20.2	OctoSPI IO Manager main features .....	568
20.3	OctoSPI IO Manager functional description .....	568
20.3.1	General description .....	568
20.3.2	OctoSPI IO Manager Matrix .....	569
20.4	OctoSPI IO Manager registers .....	570
20.4.1	OctoSPI IO Manager Port n configuration register (OCTOSPIM_PnCR) .....	570
20.4.2	OctoSPI IO Manager register map .....	572
<b>21</b>	<b>Analog-to-digital converters (ADC) .....</b>	<b>573</b>
21.1	Introduction .....	573
21.2	ADC main features .....	574
21.3	ADC implementation .....	576
21.4	ADC functional description .....	576
21.4.1	ADC block diagram .....	576
21.4.2	ADC pins and internal signals .....	578
21.4.3	Clocks .....	579
21.4.4	ADC1 connectivity .....	581
21.4.5	Slave AHB interface .....	582
21.4.6	ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN) .....	582
21.4.7	Single-ended and differential input channels .....	583
21.4.8	Calibration (ADCAL, ADCALDIF, ADCx_CALFACT) .....	583
21.4.9	ADC on-off control (ADEN, ADDIS, ADRDY) .....	586
21.4.10	Constraints when writing the ADC control bits .....	587
21.4.11	Channel selection (SQRx, JSQRx) .....	588
21.4.12	Channel-wise programmable sampling time (SMPR1, SMPR2) .....	589
21.4.13	Single conversion mode (CONT=0) .....	590
21.4.14	Continuous conversion mode (CONT=1) .....	590
21.4.15	Starting conversions (ADSTART, JADSTART) .....	591
21.4.16	ADC timing .....	592

21.4.17	Stopping an ongoing conversion (ADSTP, JADSTP) . . . . .	593
21.4.18	Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN) . . . . .	595
21.4.19	Injected channel management . . . . .	597
21.4.20	Discontinuous mode (DISCEN, DISCNUM, JDISCEN) . . . . .	598
21.4.21	Queue of context for injected conversions . . . . .	599
21.4.22	Programmable resolution (RES) - fast conversion mode . . . . .	607
21.4.23	End of conversion, end of sampling phase (EOC, JEOC, EOSMP) . . . . .	608
21.4.24	End of conversion sequence (EOS, JEOS) . . . . .	608
21.4.25	Timing diagrams example (single/continuous modes, hardware/software triggers) . . . . .	609
21.4.26	Data management . . . . .	611
21.4.27	Managing conversions using the DFSDM . . . . .	616
21.4.28	Dynamic low-power features . . . . .	617
21.4.29	Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx) . . . . .	622
21.4.30	Oversampler . . . . .	626
21.4.31	Temperature sensor . . . . .	631
21.4.32	VBAT supply monitoring . . . . .	632
21.4.33	Monitoring the internal voltage reference . . . . .	633
21.5	ADC interrupts . . . . .	635
21.6	ADC registers . . . . .	636
21.6.1	ADC interrupt and status register (ADC_ISR) . . . . .	636
21.6.2	ADC interrupt enable register (ADC_IER) . . . . .	638
21.6.3	ADC control register (ADC_CR) . . . . .	640
21.6.4	ADC configuration register (ADC_CFGR) . . . . .	643
21.6.5	ADC configuration register 2 (ADC_CFGR2) . . . . .	647
21.6.6	ADC sample time register 1 (ADC_SMPR1) . . . . .	649
21.6.7	ADC sample time register 2 (ADC_SMPR2) . . . . .	650
21.6.8	ADC watchdog threshold register 1 (ADC_TR1) . . . . .	650
21.6.9	ADC watchdog threshold register 2 (ADC_TR2) . . . . .	651
21.6.10	ADC watchdog threshold register 3 (ADC_TR3) . . . . .	652
21.6.11	ADC regular sequence register 1 (ADC_SQR1) . . . . .	653
21.6.12	ADC regular sequence register 2 (ADC_SQR2) . . . . .	654
21.6.13	ADC regular sequence register 3 (ADC_SQR3) . . . . .	655
21.6.14	ADC regular sequence register 4 (ADC_SQR4) . . . . .	656
21.6.15	ADC regular data register (ADC_DR) . . . . .	656
21.6.16	ADC injected sequence register (ADC_JSQR) . . . . .	657



21.6.17	ADC offset y register (ADC_OFy)	658
21.6.18	ADC injected channel y data register (ADC_JDRy)	659
21.6.19	ADC Analog Watchdog 2 Configuration Register (ADC_AWD2CR)	660
21.6.20	ADC Analog Watchdog 3 Configuration Register (ADC_AWD3CR)	660
21.6.21	ADC Differential mode Selection Register (ADC_DIFSEL)	661
21.6.22	ADC Calibration Factors (ADC_CALFACT)	661
21.7	ADC common registers	662
21.7.1	ADC common status register (ADC_CSR)	662
21.7.2	ADC common control register (ADC_CCR)	663
21.7.3	ADC register map	664
<b>22</b>	<b>Digital-to-analog converter (DAC)</b>	<b>668</b>
22.1	Introduction	668
22.2	DAC main features	668
22.3	DAC implementation	669
22.4	DAC functional description	670
22.4.1	DAC block diagram	670
22.4.2	DAC channel enable	671
22.4.3	DAC data format	671
22.4.4	DAC conversion	673
22.4.5	DAC output voltage	673
22.4.6	DAC trigger selection	674
22.4.7	DMA requests	675
22.4.8	Noise generation	675
22.4.9	Triangle-wave generation	676
22.4.10	DAC channel modes	677
22.4.11	DAC channel buffer calibration	680
22.4.12	Dual DAC channel conversion (if available)	682
22.5	DAC low-power modes	687
22.6	DAC interrupts	687
22.7	DAC registers	687
22.7.1	DAC control register (DAC_CR)	687
22.7.2	DAC software trigger register (DAC_SWTRGR)	690
22.7.3	DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)	691
22.7.4	DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)	692

22.7.5	DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1) . . . . .	692
22.7.6	DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2) . . . . .	692
22.7.7	DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2) . . . . .	693
22.7.8	DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2) . . . . .	693
22.7.9	Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD) . . . . .	694
22.7.10	Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD) . . . . .	694
22.7.11	Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD) . . . . .	694
22.7.12	DAC channel1 data output register (DAC_DOR1) . . . . .	695
22.7.13	DAC channel2 data output register (DAC_DOR2) . . . . .	695
22.7.14	DAC status register (DAC_SR) . . . . .	696
22.7.15	DAC calibration control register (DAC_CCR) . . . . .	697
22.7.16	DAC mode control register (DAC_MCR) . . . . .	697
22.7.17	DAC channel 1 sample and hold sample time register (DAC_SHSR1)	699
22.7.18	DAC channel 2 sample and hold sample time register (DAC_SHSR2)	699
22.7.19	DAC sample and hold time register (DAC_SHHR) . . . . .	699
22.7.20	DAC sample and hold refresh time register (DAC_SHRR) . . . . .	700
22.7.21	DAC register map . . . . .	702
<b>23</b>	<b>Digital camera interface (DCMI) . . . . .</b>	<b>704</b>
23.1	DCMI introduction . . . . .	704
23.2	DCMI main features . . . . .	704
23.3	DCMI clocks . . . . .	704
23.4	DCMI functional overview . . . . .	704
23.4.1	DCMI block diagram . . . . .	705
23.4.2	DMA interface . . . . .	705
23.4.3	DCMI physical interface . . . . .	705
23.4.4	Synchronization . . . . .	708
23.4.5	Capture modes . . . . .	710
23.4.6	Crop feature . . . . .	711
23.4.7	JPEG format . . . . .	712
23.4.8	FIFO . . . . .	712
23.5	Data format description . . . . .	713

23.5.1	Data formats	713
23.5.2	Monochrome format	713
23.5.3	RGB format	714
23.5.4	YCbCr format	714
23.5.5	YCbCr format - Y only	714
23.5.6	Half resolution image extraction	715
23.6	DCMI interrupts	715
23.7	DCMI register description	715
23.7.1	DCMI control register (DCMI_CR)	715
23.7.2	DCMI status register (DCMI_SR)	719
23.7.3	DCMI raw interrupt status register (DCMI_RIS)	720
23.7.4	DCMI interrupt enable register (DCMI_IER)	721
23.7.5	DCMI masked interrupt status register (DCMI_MIS)	722
23.7.6	DCMI interrupt clear register (DCMI_ICR)	723
23.7.7	DCMI embedded synchronization code register (DCMI_ESCR)	724
23.7.8	DCMI embedded synchronization unmask register (DCMI_ESUR)	725
23.7.9	DCMI crop window start (DCMI_CWSTRT)	726
23.7.10	DCMI crop window size (DCMI_CWSIZE)	726
23.7.11	DCMI data register (DCMI_DR)	727
23.7.12	DCMI register map	728
<b>24</b>	<b>Voltage reference buffer (VREFBUF)</b>	<b>729</b>
24.1	Introduction	729
24.2	VREFBUF functional description	729
24.3	VREFBUF registers	730
24.3.1	VREFBUF control and status register (VREFBUF_CSR)	730
24.3.2	VREFBUF calibration control register (VREFBUF_CCR)	731
24.3.3	VREFBUF register map	731
<b>25</b>	<b>Comparator (COMP)</b>	<b>732</b>
25.1	Introduction	732
25.2	COMP main features	732
25.3	COMP functional description	733
25.3.1	COMP block diagram	733
25.3.2	COMP pins and internal signals	733
25.3.3	COMP reset and clocks	734

25.3.4	Comparator LOCK mechanism	734
25.3.5	Window comparator	735
25.3.6	Hysteresis	735
25.3.7	Comparator output blanking function	736
25.3.8	COMP power and speed modes	737
25.4	COMP low-power modes	737
25.5	COMP interrupts	737
25.6	COMP registers	738
25.6.1	Comparator 1 control and status register (COMP1_CSR)	738
25.6.2	Comparator 2 control and status register (COMP2_CSR)	740
25.6.3	COMP register map	743
<b>26</b>	<b>Operational amplifiers (OPAMP)</b>	<b>744</b>
26.1	Introduction	744
26.2	OPAMP main features	744
26.3	OPAMP functional description	744
26.3.1	OPAMP reset and clocks	744
26.3.2	Initial configuration	745
26.3.3	Signal routing	745
26.3.4	OPAMP modes	746
26.3.5	Calibration	749
26.4	OPAMP low-power modes	751
26.5	OPAMP registers	752
26.5.1	OPAMP1 control/status register (OPAMP1_CSR)	752
26.5.2	OPAMP1 offset trimming register in normal mode (OPAMP1_OTR)	753
26.5.3	OPAMP1 offset trimming register in low-power mode (OPAMP1_LPOTR)	753
26.5.4	OPAMP2 control/status register (OPAMP2_CRS)	754
26.5.5	OPAMP2 offset trimming register in normal mode (OPAMP2_OTR)	755
26.5.6	OPAMP2 offset trimming register in low-power mode (OPAMP2_LPOTR)	755
26.5.7	OPAMP register map	756
<b>27</b>	<b>Digital filter for sigma delta modulators (DFSDM)</b>	<b>757</b>
27.1	Introduction	757
27.2	DFSDM main features	758
27.3	DFSDM implementation	759

27.4	DFSDM functional description	760
27.4.1	DFSDM block diagram	760
27.4.2	DFSDM pins and internal signals	761
27.4.3	DFSDM reset and clocks	762
27.4.4	Serial channel transceivers	763
27.4.5	Configuring the input serial interface	772
27.4.6	Parallel data inputs	772
27.4.7	Channel selection	774
27.4.8	Digital filter configuration	775
27.4.9	Integrator unit	776
27.4.10	Analog watchdog	777
27.4.11	Short-circuit detector	779
27.4.12	Extreme detector	780
27.4.13	Data unit block	780
27.4.14	Signed data format	781
27.4.15	Launching conversions	782
27.4.16	Continuous and fast continuous modes	782
27.4.17	Request precedence	783
27.4.18	Power optimization in run mode	784
27.5	DFSDM interrupts	784
27.6	DFSDM DMA transfer	786
27.7	DFSDM channel y registers (y=0..7)	786
27.7.1	DFSDM channel y configuration register (DFSDM_CHyCFGR1)	786
27.7.2	DFSDM channel y configuration register (DFSDM_CHyCFGR2)	788
27.7.3	DFSDM channel y analog watchdog and short-circuit detector register (DFSDM_CHyAWSCDR)	789
27.7.4	DFSDM channel y watchdog filter data register (DFSDM_CHyWDATR)	790
27.7.5	DFSDM channel y data input register (DFSDM_CHyDATINR)	790
27.7.6	DFSDM channel y delay register (DFSDM_CHyDLYR)	791
27.8	DFSDM filter x module registers (x=0..3)	792
27.8.1	DFSDM filter x control register 1 (DFSDM_FLTxCR1)	792
27.8.2	DFSDM filter x control register 2 (DFSDM_FLTxCR2)	795
27.8.3	DFSDM filter x interrupt and status register (DFSDM_FLTxISR)	796
27.8.4	DFSDM filter x interrupt flag clear register (DFSDM_FLTxICR)	798
27.8.5	DFSDM filter x injected channel group selection register (DFSDM_FLTxJCHGR)	799

27.8.6	DFSDM filter x control register (DFSDM_FLTxFCR) . . . . .	799
27.8.7	DFSDM filter x data register for injected group (DFSDM_FLTxJDATAR) . . . . .	800
27.8.8	DFSDM filter x data register for the regular channel (DFSDM_FLTxRDATAR) . . . . .	801
27.8.9	DFSDM filter x analog watchdog high threshold register (DFSDM_FLTxAWHTR) . . . . .	802
27.8.10	DFSDM filter x analog watchdog low threshold register (DFSDM_FLTxAWLTR) . . . . .	802
27.8.11	DFSDM filter x analog watchdog status register (DFSDM_FLTxAWSR) . . . . .	803
27.8.12	DFSDM filter x analog watchdog clear flag register (DFSDM_FLTxAWCFR) . . . . .	804
27.8.13	DFSDM filter x extremes detector maximum register (DFSDM_FLTxEXMAX) . . . . .	804
27.8.14	DFSDM filter x extremes detector minimum register (DFSDM_FLTxEXMIN) . . . . .	805
27.8.15	DFSDM filter x conversion timer register (DFSDM_FLTxCNVTIMR) . .	805
27.8.16	DFSDM register map . . . . .	806
<b>28</b>	<b>LCD-TFT display controller (LTDC) . . . . .</b>	<b>816</b>
28.1	Introduction . . . . .	816
28.2	LTDC main features . . . . .	816
28.3	LTDC functional description . . . . .	817
28.3.1	LTDC block diagram . . . . .	817
28.3.2	LTDC pins and external signal interface . . . . .	817
28.3.3	LTDC reset and clocks . . . . .	818
28.4	LTDC programmable parameters . . . . .	819
28.4.1	LTDC global configuration parameters . . . . .	819
28.4.2	Layer programmable parameters . . . . .	822
28.5	LTDC interrupts . . . . .	826
28.6	LTDC programming procedure . . . . .	827
28.7	LTDC registers . . . . .	829
28.7.1	LTDC synchronization size configuration register (LTDC_SSCR) . . . .	829
28.7.2	LTDC back porch configuration register (LTDC_BPCR) . . . . .	829
28.7.3	LTDC active width configuration register (LTDC_AWCR) . . . . .	830
28.7.4	LTDC total width configuration register (LTDC_TWCR) . . . . .	831
28.7.5	LTDC global control register (LTDC_GCR) . . . . .	831
28.7.6	LTDC shadow reload configuration register (LTDC_SRCR) . . . . .	833

28.7.7	LTDC background color configuration register (LTDC_BCCR) . . . . .	833
28.7.8	LTDC interrupt enable register (LTDC_IER) . . . . .	834
28.7.9	LTDC interrupt status register (LTDC_ISR) . . . . .	835
28.7.10	LTDC Interrupt Clear Register (LTDC_ICR) . . . . .	835
28.7.11	LTDC line interrupt position configuration register (LTDC_LIPCR) . . .	836
28.7.12	LTDC current position status register (LTDC_CPSR) . . . . .	836
28.7.13	LTDC current display status register (LTDC_CDSR) . . . . .	837
28.7.14	LTDC layer x control register (LTDC_LxCR) . . . . .	838
28.7.15	LTDC layer x window horizontal position configuration register (LTDC_LxWHPCR) . . . . .	839
28.7.16	LTDC layer x window vertical position configuration register (LTDC_LxWVPCR) . . . . .	840
28.7.17	LTDC layer x color keying configuration register (LTDC_LxCKCR) . . . . .	841
28.7.18	LTDC layer x pixel format configuration register (LTDC_LxPFCR) . . . . .	841
28.7.19	LTDC layer x constant alpha configuration register (LTDC_LxCACR) . . . . .	842
28.7.20	LTDC layer x default color configuration register (LTDC_LxDCCR) . . . . .	843
28.7.21	LTDC layer x blending factors configuration register (LTDC_LxBFCR) . . . . .	844
28.7.22	LTDC layer x color frame buffer address register (LTDC_LxCFBAR) . . . . .	845
28.7.23	LTDC layer x color frame buffer length register (LTDC_LxCFBLR) . . . . .	845
28.7.24	LTDC layer x color frame buffer line number register (LTDC_LxCFBLNR) . . . . .	846
28.7.25	LTDC layer x CLUT write register (LTDC_LxCLUTWR) . . . . .	847
28.7.26	LTDC register map . . . . .	848
<b>29</b>	<b>DSI Host (DSIHOST) . . . . .</b>	<b>851</b>
29.1	Introduction . . . . .	851
29.2	Standard and references . . . . .	851
29.3	DSI Host main features . . . . .	852
29.4	DSI Host functional description . . . . .	853
29.4.1	General description . . . . .	853
29.4.2	Supported resolutions and frame rates . . . . .	853
29.4.3	System level architecture . . . . .	854

- 29.5 Functional description: Video mode on LTDC interface ..... 856
  - 29.5.1 Video transmission mode ..... 857
  - 29.5.2 Updating the LTDC interface configuration in video mode ..... 859
- 29.6 Functional description: adapted command mode on LTDC interface ... 861
- 29.7 Functional description: APB slave generic interface ..... 865
  - 29.7.1 Packet transmission using the generic interface ..... 866
- 29.8 Functional description: Timeout counters ..... 869
  - 29.8.1 Contention error detection timeout counters ..... 869
  - 29.8.2 Peripheral response timeout counters ..... 870
- 29.9 Functional description: transmission of commands ..... 875
  - 29.9.1 Transmission of commands in Video mode ..... 875
  - 29.9.2 Transmission of commands in Low-power mode ..... 877
  - 29.9.3 Transmission of commands in High-speed ..... 881
  - 29.9.4 Read command transmission ..... 881
  - 29.9.5 Clock lane in Low-power mode ..... 882
- 29.10 Functional description: virtual channels ..... 884
- 29.11 Functional description: video mode pattern generator ..... 885
  - 29.11.1 Color bar pattern ..... 885
  - 29.11.2 Color coding ..... 886
  - 29.11.3 BER testing pattern ..... 887
  - 29.11.4 Video mode pattern generator resolution ..... 888
- 29.12 Functional description: D-PHY management ..... 889
  - 29.12.1 D-PHY configuration ..... 889
  - 29.12.2 Special D-PHY operations ..... 891
  - 29.12.3 Special Low-power D-PHY functions ..... 891
  - 29.12.4 DSI PLL control ..... 892
  - 29.12.5 Regulator control ..... 893
- 29.13 Functional description: interrupts and errors ..... 894
  - 29.13.1 DSI wrapper interrupts ..... 894
  - 29.13.2 DSI host interrupts and errors ..... 894
- 29.14 Programing procedure ..... 901
  - 29.14.1 Programing procedure overview ..... 901
  - 29.14.2 Configuring the D-PHY parameters ..... 901
  - 29.14.3 Configuring the DSI host timing ..... 902
  - 29.14.4 Configuring flow control and DBI interface ..... 903
  - 29.14.5 Configuring the DSI host LTDC interface ..... 903



29.14.6	Configuring the video mode	904
29.14.7	Configuring the adapted command mode	907
29.14.8	Configuring the video mode pattern generator	908
29.14.9	Managing ULPM	910
29.15	DSI Host registers	912
29.15.1	DSI Host Version Register (DSI_VR)	912
29.15.2	DSI Host Control Register (DSI_CR)	912
29.15.3	DSI HOST Clock Control Register (DSI_CCR)	913
29.15.4	DSI Host LTDC VCID Register (DSI_LVCIDR)	913
29.15.5	DSI Host LTDC Color Coding Register (DSI_LCOLCR)	914
29.15.6	DSI Host LTDC Polarity Configuration Register (DSI_LPCR)	914
29.15.7	DSI Host Low-Power mode Configuration Register (DSI_LPMCR)	915
29.15.8	DSI Host Protocol Configuration Register (DSI_PCR)	915
29.15.9	DSI Host Generic VCID Register (DSI_GVCIDR)	916
29.15.10	DSI Host mode Configuration Register (DSI_MCR)	917
29.15.11	DSI Host video mode Configuration Register (DSI_VMCR)	917
29.15.12	DSI Host video Packet Configuration Register (DSI_VPCR)	919
29.15.13	DSI Host video Chunks Configuration Register (DSI_VCCR)	919
29.15.14	DSI Host Video Null Packet Configuration Register (DSI_VNPCR)	920
29.15.15	DSI Host Video HSA Configuration Register (DSI_VHSACR)	920
29.15.16	DSI Host Video HBP Configuration Register (DSI_VHBPCR)	921
29.15.17	DSI Host Video Line Configuration Register (DSI_VLCR)	921
29.15.18	DSI Host Video VSA Configuration Register (DSI_VVSACR)	921
29.15.19	DSI Host Video VBP Configuration Register (DSI_VVBPCR)	922
29.15.20	DSI Host Video VFP Configuration Register (DSI_VVFPCR)	922
29.15.21	DSI Host Video VA Configuration Register (DSI_VVACR)	923
29.15.22	DSI Host LTDC Command Configuration Register (DSI_LCCR)	923
29.15.23	DSI Host Command mode Configuration Register (DSI_CMCR)	924
29.15.24	DSI Host Generic Header Configuration Register (DSI_GHCR)	926
29.15.25	DSI Host Generic Payload Data Register (DSI_GPDR)	926
29.15.26	DSI Host Generic Packet Status Register (DSI_GPSR)	927
29.15.27	DSI Host Timeout Counter Configuration Register 0 (DSI_TCCR0)	928
29.15.28	DSI Host Timeout Counter Configuration Register 1 (DSI_TCCR1)	928
29.15.29	DSI Host Timeout Counter Configuration Register 2 (DSI_TCCR2)	929
29.15.30	DSI Host Timeout Counter Configuration Register 3 (DSI_TCCR3)	929
29.15.31	DSI Host Timeout Counter Configuration Register 4 (DSI_TCCR4)	930
29.15.32	DSI Host Timeout Counter Configuration Register 5 (DSI_TCCR5)	931

29.15.33 DSI Host Clock Lane Configuration Register (DSI\_CLCR) . . . . . 931

29.15.34 DSI Host Clock Lane Timer Configuration Register (DSI\_CLTCR) . . . 932

29.15.35 DSI Host Data Lane Timer Configuration Register (DSI\_DLTCR) . . . . 932

29.15.36 DSI Host PHY Control Register (DSI\_PCTLR) . . . . . 933

29.15.37 DSI Host PHY Configuration Register (DSI\_PCONFR) . . . . . 933

29.15.38 DSI Host PHY ULPS Control Register (DSI\_PUCR) . . . . . 934

29.15.39 DSI Host PHY TX Triggers Configuration Register (DSI\_PTTCR) . . . 935

29.15.40 DSI Host PHY Status Register (DSI\_PSR) . . . . . 935

29.15.41 DSI Host Interrupt & Status Register 0 (DSI\_ISR0) . . . . . 936

29.15.42 DSI Host Interrupt & Status Register 1 (DSI\_ISR1) . . . . . 937

29.15.43 DSI Host Interrupt Enable Register 0 (DSI\_IER0) . . . . . 938

29.15.44 DSI Host Interrupt Enable Register 1 (DSI\_IER1) . . . . . 941

29.15.45 DSI Host Force Interrupt Register 0 (DSI\_FIR0) . . . . . 942

29.15.46 DSI Host Force Interrupt Register 1 (DSI\_FIR1) . . . . . 944

29.15.47 DSI Host Video Shadow Control Register (DSI\_VSCR) . . . . . 945

29.15.48 DSI Host LTDC Current VCID Register (DSI\_LCVCIDR) . . . . . 945

29.15.49 DSI Host LTDC Current Color Coding Register (DSI\_LCCCR) . . . . . 946

29.15.50 DSI Host Low-Power mode Current Configuration Register  
(DSI\_LPMCCR) . . . . . 946

29.15.51 DSI Host Video mode Current Configuration Register  
(DSI\_VMCCR) . . . . . 947

29.15.52 DSI Host Video Packet Current Configuration Register  
(DSI\_VPCCR) . . . . . 948

29.15.53 DSI Host Video Chunks Current Configuration Register  
(DSI\_VCCCR) . . . . . 949

29.15.54 DSI Host Video Null Packet Current Configuration Register  
(DSI\_VNPCCR) . . . . . 949

29.15.55 DSI Host Video HSA Current Configuration Register  
(DSI\_VHSACCR) . . . . . 950

29.15.56 DSI Host Video HBP Current Configuration Register  
(DSI\_VHBPCR) . . . . . 950

29.15.57 DSI Host Video Line Current Configuration Register (DSI\_VLCCR) . . 951

29.15.58 DSI Host Video VSA Current Configuration Register  
(DSI\_VVSACCR) . . . . . 951

29.15.59 DSI Host Video VBP Current Configuration Register  
(DSI\_VVBPCR) . . . . . 952

29.15.60 DSI Host Video VFP Current Configuration Register  
(DSI\_VVFPCCR) . . . . . 952

29.15.61 DSI Host Video VA Current Configuration Register  
(DSI\_VVACCR) . . . . . 953

29.16	DSI Wrapper Registers	954
29.16.1	DSI Wrapper Configuration Register (DSI_WCFGR)	954
29.16.2	DSI Wrapper Control Register (DSI_WCR)	955
29.16.3	DSI Wrapper Interrupt Enable Register (DSI_WIER)	956
29.16.4	DSI Wrapper Interrupt & Status Register (DSI_WISR)	957
29.16.5	DSI Wrapper Interrupt Flag Clear Register (DSI_WIFCR)	958
29.16.6	DSI Wrapper PHY Configuration Register 0 (DSI_WPCR0)	959
29.16.7	DSI Wrapper PHY Configuration Register 1 (DSI_WPCR1)	961
29.16.8	DSI Wrapper PHY Configuration Register 2 (DSI_WPCR2)	963
29.16.9	DSI Wrapper PHY Configuration Register 3 (DSI_WPCR4)	964
29.16.10	DSI Wrapper PHY Configuration Register 4 (DSI_WPCR4)	965
29.16.11	DSI Wrapper Regulator and PLL Control Register (DSI_WRPCR)	965
29.17	DSI Host register map	967
<b>30</b>	<b>Touch sensing controller (TSC)</b>	<b>972</b>
30.1	Introduction	972
30.2	TSC main features	972
30.3	TSC functional description	973
30.3.1	TSC block diagram	973
30.3.2	Surface charge transfer acquisition overview	973
30.3.3	Reset and clocks	975
30.3.4	Charge transfer acquisition sequence	976
30.3.5	Spread spectrum feature	977
30.3.6	Max count error	977
30.3.7	Sampling capacitor I/O and channel I/O mode selection	978
30.3.8	Acquisition mode	979
30.3.9	I/O hysteresis and analog switch control	979
30.4	TSC low-power modes	980
30.5	TSC interrupts	980
30.6	TSC registers	981
30.6.1	TSC control register (TSC_CR)	981
30.6.2	TSC interrupt enable register (TSC_IER)	983
30.6.3	TSC interrupt clear register (TSC_ICR)	984
30.6.4	TSC interrupt status register (TSC_ISR)	985
30.6.5	TSC I/O hysteresis control register (TSC_IOHCR)	985
30.6.6	TSC I/O analog switch control register (TSC_IOASCR)	986

30.6.7	TSC I/O sampling control register (TSC_IOSCR) . . . . .	986
30.6.8	TSC I/O channel control register (TSC_IOCCR) . . . . .	987
30.6.9	TSC I/O group control status register (TSC_IOGCSR) . . . . .	987
30.6.10	TSC I/O group x counter register (TSC_IOGxCR) . . . . .	988
30.6.11	TSC register map . . . . .	989
<b>31</b>	<b>True random number generator (RNG) . . . . .</b>	<b>991</b>
31.1	Introduction . . . . .	991
31.2	RNG main features . . . . .	991
31.3	RNG functional description . . . . .	992
31.3.1	RNG block diagram . . . . .	992
31.3.2	RNG internal signals . . . . .	992
31.3.3	Random number generation . . . . .	993
31.3.4	RNG initialization . . . . .	995
31.3.5	RNG operation . . . . .	996
31.3.6	RNG clocking . . . . .	997
31.3.7	Error management . . . . .	997
31.4	RNG low-power usage . . . . .	998
31.5	RNG interrupts . . . . .	999
31.6	RNG processing time . . . . .	999
31.7	Entropy source validation . . . . .	999
31.7.1	Introduction . . . . .	999
31.7.2	Validation conditions . . . . .	1000
31.7.3	Data collection . . . . .	1000
31.8	RNG registers . . . . .	1001
31.8.1	RNG control register (RNG_CR) . . . . .	1001
31.8.2	RNG status register (RNG_SR) . . . . .	1002
31.8.3	RNG data register (RNG_DR) . . . . .	1003
31.8.4	RNG register map . . . . .	1004
<b>32</b>	<b>AES hardware accelerator (AES) . . . . .</b>	<b>1005</b>
32.1	Introduction . . . . .	1005
32.2	AES main features . . . . .	1005
32.3	AES implementation . . . . .	1006
32.4	AES functional description . . . . .	1006
32.4.1	AES block diagram . . . . .	1006

32.4.2	AES internal signals	1006
32.4.3	AES cryptographic core	1007
32.4.4	AES procedure to perform a cipher operation	1012
32.4.5	AES decryption key preparation	1016
32.4.6	AES ciphertext stealing and data padding	1017
32.4.7	AES task suspend and resume	1018
32.4.8	AES basic chaining modes (ECB, CBC)	1018
32.4.9	AES counter (CTR) mode	1023
32.4.10	AES Galois/counter mode (GCM)	1026
32.4.11	AES Galois message authentication code (GMAC)	1031
32.4.12	AES counter with CBC-MAC (CCM)	1033
32.4.13	AES data registers and data swapping	1039
32.4.14	AES key registers	1041
32.4.15	AES initialization vector registers	1041
32.4.16	AES DMA interface	1041
32.4.17	AES error management	1044
32.5	AES interrupts	1044
32.6	AES processing latency	1045
32.7	AES registers	1046
32.7.1	AES control register (AES_CR)	1046
32.7.2	AES status register (AES_SR)	1048
32.7.3	AES data input register (AES_DINR)	1050
32.7.4	AES data output register (AES_DOUTR)	1051
32.7.5	AES key register 0 (AES_KEYR0)	1051
32.7.6	AES key register 1 (AES_KEYR1)	1052
32.7.7	AES key register 2 (AES_KEYR2)	1052
32.7.8	AES key register 3 (AES_KEYR3)	1052
32.7.9	AES initialization vector register 0 (AES_IVR0)	1053
32.7.10	AES initialization vector register 1 (AES_IVR1)	1053
32.7.11	AES initialization vector register 2 (AES_IVR2)	1053
32.7.12	AES initialization vector register 3 (AES_IVR3)	1054
32.7.13	AES key register 4 (AES_KEYR4)	1054
32.7.14	AES key register 5 (AES_KEYR5)	1054
32.7.15	AES key register 6 (AES_KEYR6)	1055
32.7.16	AES key register 7 (AES_KEYR7)	1055
32.7.17	AES suspend registers (AES_SUSPxR)	1055
32.7.18	AES register map	1057

<b>33</b>	<b>Hash processor (HASH)</b>	<b>1059</b>
33.1	Introduction	1059
33.2	HASH main features	1059
33.3	HASH functional description	1060
33.3.1	HASH block diagram	1060
33.3.2	HASH internal signals	1060
33.3.3	About secure hash algorithms	1061
33.3.4	Message data feeding	1061
33.3.5	Message digest computing	1063
33.3.6	Message padding	1064
33.3.7	HMAC operation	1065
33.3.8	Context swapping	1067
33.3.9	HASH DMA interface	1069
33.3.10	HASH error management	1069
33.4	HASH interrupts	1069
33.5	HASH processing time	1070
33.6	HASH registers	1071
33.6.1	HASH control register (HASH_CR)	1071
33.6.2	HASH data input register (HASH_DIN)	1074
33.6.3	HASH start register (HASH_STR)	1075
33.6.4	HASH digest registers (HASH_HR0..7)	1076
33.6.5	HASH interrupt enable register (HASH_IMR)	1079
33.6.6	HASH status register (HASH_SR)	1080
33.6.7	HASH context swap registers (HASH_CSRx)	1081
33.6.8	HASH register map	1082
<b>34</b>	<b>Advanced-control timers (TIM1/TIM8)</b>	<b>1083</b>
34.1	TIM1/TIM8 introduction	1083
34.2	TIM1/TIM8 main features	1083
34.3	TIM1/TIM8 functional description	1085
34.3.1	Time-base unit	1085
34.3.2	Counter modes	1087
34.3.3	Repetition counter	1098
34.3.4	External trigger input	1100
34.3.5	Clock selection	1101
34.3.6	Capture/compare channels	1105

34.3.7	Input capture mode	1107
34.3.8	PWM input mode	1108
34.3.9	Forced output mode	1109
34.3.10	Output compare mode	1110
34.3.11	PWM mode	1111
34.3.12	Asymmetric PWM mode	1114
34.3.13	Combined PWM mode	1115
34.3.14	Combined 3-phase PWM mode	1116
34.3.15	Complementary outputs and dead-time insertion	1117
34.3.16	Using the break function	1119
34.3.17	Bidirectional break inputs	1125
34.3.18	Clearing the OCxREF signal on an external event	1126
34.3.19	6-step PWM generation	1128
34.3.20	One-pulse mode	1129
34.3.21	Retriggerable one pulse mode (OPM)	1130
34.3.22	Encoder interface mode	1131
34.3.23	UIF bit remapping	1133
34.3.24	Timer input XOR function	1134
34.3.25	Interfacing with Hall sensors	1134
34.3.26	Timer synchronization	1137
34.3.27	ADC synchronization	1141
34.3.28	DMA burst mode	1141
34.3.29	Debug mode	1142
34.4	TIM1/TIM8 registers	1143
34.4.1	TIM1/TIM8 control register 1 (TIMx_CR1)	1143
34.4.2	TIM1/TIM8 control register 2 (TIMx_CR2)	1144
34.4.3	TIM1/TIM8 slave mode control register (TIMx_SMCR)	1147
34.4.4	TIM1/TIM8 DMA/interrupt enable register (TIMx_DIER)	1149
34.4.5	TIM1/TIM8 status register (TIMx_SR)	1151
34.4.6	TIM1/TIM8 event generation register (TIMx_EGR)	1153
34.4.7	TIM1/TIM8 capture/compare mode register 1 (TIMx_CCMR1)	1154
34.4.8	TIM1/TIM8 capture/compare mode register 2 (TIMx_CCMR2)	1158
34.4.9	TIM1/TIM8 capture/compare enable register (TIMx_CCER)	1160
34.4.10	TIM1/TIM8 counter (TIMx_CNT)	1164
34.4.11	TIM1/TIM8 prescaler (TIMx_PSC)	1164
34.4.12	TIM1/TIM8 auto-reload register (TIMx_ARR)	1164
34.4.13	TIM1/TIM8 repetition counter register (TIMx_RCR)	1165

- 34.4.14 TIM1/TIM8 capture/compare register 1 (TIMx\_CCR1) ..... 1165
- 34.4.15 TIM1/TIM8 capture/compare register 2 (TIMx\_CCR2) ..... 1166
- 34.4.16 TIM1/TIM8 capture/compare register 3 (TIMx\_CCR3) ..... 1166
- 34.4.17 TIM1/TIM8 capture/compare register 4 (TIMx\_CCR4) ..... 1167
- 34.4.18 TIM1/TIM8 break and dead-time register (TIMx\_BDTR) ..... 1167
- 34.4.19 TIM1/TIM8 DMA control register (TIMx\_DCR) ..... 1171
- 34.4.20 TIM1/TIM8 DMA address for full transfer (TIMx\_DMAR) ..... 1172
- 34.4.21 TIM1 option register 1 (TIM1\_OR1) ..... 1173
- 34.4.22 TIM8 option register 1 (TIM8\_OR1) ..... 1173
- 34.4.23 TIM1/TIM8 capture/compare mode register 3 (TIMx\_CCMR3) ..... 1174
- 34.4.24 TIM1/TIM8 capture/compare register 5 (TIMx\_CCR5) ..... 1175
- 34.4.25 TIM1/TIM8 capture/compare register 6 (TIMx\_CCR6) ..... 1176
- 34.4.26 TIM1 option register 2 (TIM1\_OR2) ..... 1176
- 34.4.27 TIM1 option register 3 (TIM1\_OR3) ..... 1178
- 34.4.28 TIM8 option register 2 (TIM8\_OR2) ..... 1179
- 34.4.29 TIM8 option register 3 (TIM8\_OR3) ..... 1181
- 34.4.30 TIM1 register map ..... 1183
- 34.4.31 TIM8 register map ..... 1186

**35 General-purpose timers (TIM2/TIM3/TIM4/TIM5) ..... 1189**

- 35.1 TIM2/TIM3/TIM4/TIM5 introduction ..... 1189
- 35.2 TIM2/TIM3/TIM4/TIM5 main features ..... 1189
- 35.3 TIM2/TIM3/TIM4/TIM5 functional description ..... 1191
  - 35.3.1 Time-base unit ..... 1191
  - 35.3.2 Counter modes ..... 1193
  - 35.3.3 Clock selection ..... 1203
  - 35.3.4 Capture/Compare channels ..... 1207
  - 35.3.5 Input capture mode ..... 1209
  - 35.3.6 PWM input mode ..... 1210
  - 35.3.7 Forced output mode ..... 1211
  - 35.3.8 Output compare mode ..... 1212
  - 35.3.9 PWM mode ..... 1213
  - 35.3.10 Asymmetric PWM mode ..... 1216
  - 35.3.11 Combined PWM mode ..... 1217
  - 35.3.12 Clearing the OCxREF signal on an external event ..... 1218
  - 35.3.13 One-pulse mode ..... 1220
  - 35.3.14 Retriggerable one pulse mode (OPM) ..... 1221



35.3.15	Encoder interface mode	1222
35.3.16	UIF bit remapping	1224
35.3.17	Timer input XOR function	1224
35.3.18	Timers and external trigger synchronization	1225
35.3.19	Timer synchronization	1228
35.3.20	DMA burst mode	1232
35.3.21	Debug mode	1233
<b>35.4</b>	<b>TIM2/TIM3/TIM4/TIM5 registers</b>	<b>1234</b>
35.4.1	TIMx control register 1 (TIMx_CR1)	1234
35.4.2	TIMx control register 2 (TIMx_CR2)	1235
35.4.3	TIMx slave mode control register (TIMx_SMCR)	1237
35.4.4	TIMx DMA/Interrupt enable register (TIMx_DIER)	1240
35.4.5	TIMx status register (TIMx_SR)	1241
35.4.6	TIMx event generation register (TIMx_EGR)	1242
35.4.7	TIMx capture/compare mode register 1 (TIMx_CCMR1)	1243
35.4.8	TIMx capture/compare mode register 2 (TIMx_CCMR2)	1247
35.4.9	TIMx capture/compare enable register (TIMx_CCER)	1249
35.4.10	TIMx counter (TIMx_CNT)	1250
35.4.11	TIMx prescaler (TIMx_PSC)	1251
35.4.12	TIMx auto-reload register (TIMx_ARR)	1251
35.4.13	TIMx capture/compare register 1 (TIMx_CCR1)	1251
35.4.14	TIMx capture/compare register 2 (TIMx_CCR2)	1252
35.4.15	TIMx capture/compare register 3 (TIMx_CCR3)	1252
35.4.16	TIMx capture/compare register 4 (TIMx_CCR4)	1253
35.4.17	TIMx DMA control register (TIMx_DCR)	1254
35.4.18	TIMx DMA address for full transfer (TIMx_DMAR)	1254
35.4.19	TIM2 option register 1 (TIM2_OR1)	1255
35.4.20	TIM3 option register 1 (TIM3_OR1)	1255
35.4.21	TIM2 option register 2 (TIM2_OR2)	1256
35.4.22	TIM3 option register 2 (TIM3_OR2)	1256
35.4.23	TIMx register map	1257
<b>36</b>	<b>General-purpose timers (TIM15/TIM16/TIM17)</b>	<b>1260</b>
36.1	TIM15/TIM16/TIM17 introduction	1260
36.2	TIM15 main features	1260
36.3	TIM16/TIM17 main features	1261
36.4	TIM15/TIM16/TIM17 functional description	1264

36.4.1	Time-base unit	1264
36.4.2	Counter modes	1266
36.4.3	Repetition counter	1270
36.4.4	Clock selection	1271
36.4.5	Capture/compare channels	1273
36.4.6	Input capture mode	1276
36.4.7	PWM input mode (only for TIM15)	1277
36.4.8	Forced output mode	1278
36.4.9	Output compare mode	1278
36.4.10	PWM mode	1279
36.4.11	Combined PWM mode (TIM15 only)	1280
36.4.12	Complementary outputs and dead-time insertion	1282
36.4.13	Using the break function	1284
36.4.14	Bidirectional break inputs	1288
36.4.15	One-pulse mode	1290
36.4.16	Retriggerable one pulse mode (OPM) (TIM15 only)	1291
36.4.17	UIF bit remapping	1292
36.4.18	Timer input XOR function (TIM15 only)	1293
36.4.19	External trigger synchronization (TIM15 only)	1294
36.4.20	Slave mode – combined reset + trigger mode	1296
36.4.21	DMA burst mode	1296
36.4.22	Timer synchronization (TIM15)	1298
36.4.23	Debug mode	1298
36.5	TIM15 registers	1299
36.5.1	TIM15 control register 1 (TIM15_CR1)	1299
36.5.2	TIM15 control register 2 (TIM15_CR2)	1300
36.5.3	TIM15 slave mode control register (TIM15_SMCR)	1302
36.5.4	TIM15 DMA/interrupt enable register (TIM15_DIER)	1303
36.5.5	TIM15 status register (TIM15_SR)	1304
36.5.6	TIM15 event generation register (TIM15_EGR)	1306
36.5.7	TIM15 capture/compare mode register 1 (TIM15_CCMR1)	1307
36.5.8	TIM15 capture/compare enable register (TIM15_CCER)	1311
36.5.9	TIM15 counter (TIM15_CNT)	1314
36.5.10	TIM15 prescaler (TIM15_PSC)	1314
36.5.11	TIM15 auto-reload register (TIM15_ARR)	1314
36.5.12	TIM15 repetition counter register (TIM15_RCR)	1315
36.5.13	TIM15 capture/compare register 1 (TIM15_CCR1)	1315

36.5.14	TIM15 capture/compare register 2 (TIM15_CCR2)	1316
36.5.15	TIM15 break and dead-time register (TIM15_BDTR)	1316
36.5.16	TIM15 DMA control register (TIM15_DCR)	1319
36.5.17	TIM15 DMA address for full transfer (TIM15_DMAR)	1319
36.5.18	TIM15 option register 1 (TIM15_OR1)	1320
36.5.19	TIM15 option register 2 (TIM15_OR2)	1320
36.5.20	TIM15 register map	1322
<b>36.6</b>	<b>TIM16/TIM17 registers</b>	<b>1325</b>
36.6.1	TIM16/TIM17 control register 1 (TIMx_CR1)	1325
36.6.2	TIM16/TIM17 control register 2 (TIMx_CR2)	1326
36.6.3	TIM16/TIM17 DMA/interrupt enable register (TIMx_DIER)	1327
36.6.4	TIM16/TIM17 status register (TIMx_SR)	1328
36.6.5	TIM16/TIM17 event generation register (TIMx_EGR)	1329
36.6.6	TIM16/TIM17 capture/compare mode register 1 (TIMx_CCMR1)	1330
36.6.7	TIM16/TIM17 capture/compare enable register (TIMx_CCER)	1332
36.6.8	TIM16/TIM17 counter (TIMx_CNT)	1334
36.6.9	TIM16/TIM17 prescaler (TIMx_PSC)	1335
36.6.10	TIM16/TIM17 auto-reload register (TIMx_ARR)	1335
36.6.11	TIM16/TIM17 repetition counter register (TIMx_RCR)	1336
36.6.12	TIM16/TIM17 capture/compare register 1 (TIMx_CCR1)	1336
36.6.13	TIM16/TIM17 break and dead-time register (TIMx_BDTR)	1337
36.6.14	TIM16/TIM17 DMA control register (TIMx_DCR)	1339
36.6.15	TIM16/TIM17 DMA address for full transfer (TIMx_DMAR)	1340
36.6.16	TIM16 option register 1 (TIM16_OR1)	1340
36.6.17	TIM16 option register 2 (TIM16_OR2)	1341
36.6.18	TIM17 option register 1 (TIM17_OR1)	1342
36.6.19	TIM17 option register 2 (TIM17_OR2)	1343
36.6.20	TIM16/TIM17 register map	1345
<b>37</b>	<b>Basic timers (TIM6/TIM7)</b>	<b>1347</b>
37.1	TIM6/TIM7 introduction	1347
37.2	TIM6/TIM7 main features	1347
37.3	TIM6/TIM7 functional description	1348
37.3.1	Time-base unit	1348
37.3.2	Counting mode	1350
37.3.3	UIF bit remapping	1353
37.3.4	Clock source	1353

37.3.5	Debug mode	1354
<b>37.4</b>	<b>TIM6/TIM7 registers</b>	<b>1354</b>
37.4.1	TIM6/TIM7 control register 1 (TIMx_CR1)	1354
37.4.2	TIM6/TIM7 control register 2 (TIMx_CR2)	1356
37.4.3	TIM6/TIM7 DMA/Interrupt enable register (TIMx_DIER)	1356
37.4.4	TIM6/TIM7 status register (TIMx_SR)	1357
37.4.5	TIM6/TIM7 event generation register (TIMx_EGR)	1357
37.4.6	TIM6/TIM7 counter (TIMx_CNT)	1357
37.4.7	TIM6/TIM7 prescaler (TIMx_PSC)	1358
37.4.8	TIM6/TIM7 auto-reload register (TIMx_ARR)	1358
37.4.9	TIM6/TIM7 register map	1359
<b>38</b>	<b>Low-power timer (LPTIM)</b>	<b>1360</b>
38.1	Introduction	1360
38.2	LPTIM main features	1360
38.3	LPTIM implementation	1360
38.4	LPTIM functional description	1361
38.4.1	LPTIM block diagram	1361
38.4.2	LPTIM trigger mapping	1361
38.4.3	LPTIM reset and clocks	1362
38.4.4	Glitch filter	1362
38.4.5	Prescaler	1363
38.4.6	Trigger multiplexer	1364
38.4.7	Operating mode	1364
38.4.8	Timeout function	1366
38.4.9	Waveform generation	1366
38.4.10	Register update	1367
38.4.11	Counter mode	1368
38.4.12	Timer enable	1368
38.4.13	Timer counter reset	1369
38.4.14	Encoder mode	1369
38.5	LPTIM low power modes	1371
38.6	LPTIM interrupts	1372
38.7	LPTIM registers	1372
38.7.1	LPTIM interrupt and status register (LPTIM_ISR)	1372
38.7.2	LPTIM interrupt clear register (LPTIM_ICR)	1373

38.7.3	LPTIM interrupt enable register (LPTIM_IER) .....	1374
38.7.4	LPTIM configuration register (LPTIM_CFGR) .....	1375
38.7.5	LPTIM control register (LPTIM_CR) .....	1378
38.7.6	LPTIM compare register (LPTIM_CMP) .....	1379
38.7.7	LPTIM autoreload register (LPTIM_ARR) .....	1380
38.7.8	LPTIM counter register (LPTIM_CNT) .....	1380
38.7.9	LPTIM1 option register (LPTIM1_OR) .....	1381
38.7.10	LPTIM2 option register (LPTIM2_OR) .....	1381
38.7.11	LPTIM register map .....	1382
<b>39</b>	<b>Infrared interface (IRTIM) .....</b>	<b>1384</b>
<b>40</b>	<b>Independent watchdog (IWDG) .....</b>	<b>1385</b>
40.1	Introduction .....	1385
40.2	IWDG main features .....	1385
40.3	IWDG functional description .....	1385
40.3.1	IWDG block diagram .....	1385
40.3.2	Window option .....	1386
40.3.3	Hardware watchdog .....	1387
40.3.4	Low-power freeze .....	1387
40.3.5	Behavior in Stop and Standby modes .....	1387
40.3.6	Register access protection .....	1387
40.3.7	Debug mode .....	1387
40.4	IWDG registers .....	1388
40.4.1	Key register (IWDG_KR) .....	1388
40.4.2	Prescaler register (IWDG_PR) .....	1389
40.4.3	Reload register (IWDG_RLR) .....	1390
40.4.4	Status register (IWDG_SR) .....	1391
40.4.5	Window register (IWDG_WINR) .....	1392
40.4.6	IWDG register map .....	1393
<b>41</b>	<b>System window watchdog (WWDG) .....</b>	<b>1394</b>
41.1	Introduction .....	1394
41.2	WWDG main features .....	1394
41.3	WWDG functional description .....	1394
41.3.1	WWDG block diagram .....	1395
41.3.2	Enabling the watchdog .....	1395

41.3.3	Controlling the downcounter	1395
41.3.4	Advanced watchdog interrupt feature	1395
41.3.5	How to program the watchdog timeout	1396
41.3.6	Debug mode	1397
41.4	WWDG registers	1398
41.4.1	Control register (WWDG_CR)	1398
41.4.2	Configuration register (WWDG_CFR)	1398
41.4.3	Status register (WWDG_SR)	1399
41.4.4	WWDG register map	1400
<b>42</b>	<b>Real-time clock (RTC)</b>	<b>1401</b>
42.1	Introduction	1401
42.2	RTC main features	1402
42.3	RTC functional description	1403
42.3.1	RTC block diagram	1403
42.3.2	GPIOs controlled by the RTC	1404
42.3.3	Clock and prescalers	1406
42.3.4	Real-time clock and calendar	1407
42.3.5	Programmable alarms	1407
42.3.6	Periodic auto-wakeup	1407
42.3.7	RTC initialization and configuration	1408
42.3.8	Reading the calendar	1410
42.3.9	Resetting the RTC	1411
42.3.10	RTC synchronization	1411
42.3.11	RTC reference clock detection	1412
42.3.12	RTC smooth digital calibration	1413
42.3.13	Time-stamp function	1415
42.3.14	Tamper detection	1415
42.3.15	Calibration clock output	1417
42.3.16	Alarm output	1418
42.4	RTC low-power modes	1418
42.5	RTC interrupts	1419
42.6	RTC registers	1420
42.6.1	RTC time register (RTC_TR)	1420
42.6.2	RTC date register (RTC_DR)	1421
42.6.3	RTC control register (RTC_CR)	1422

42.6.4	RTC initialization and status register (RTC_ISR) . . . . .	1425
42.6.5	RTC prescaler register (RTC_PRER) . . . . .	1428
42.6.6	RTC wakeup timer register (RTC_WUTR) . . . . .	1429
42.6.7	RTC alarm A register (RTC_ALRMAR) . . . . .	1430
42.6.8	RTC alarm B register (RTC_ALRMBR) . . . . .	1431
42.6.9	RTC write protection register (RTC_WPR) . . . . .	1432
42.6.10	RTC sub second register (RTC_SSR) . . . . .	1432
42.6.11	RTC shift control register (RTC_SHIFTR) . . . . .	1433
42.6.12	RTC timestamp time register (RTC_TSTR) . . . . .	1434
42.6.13	RTC timestamp date register (RTC_TSDR) . . . . .	1435
42.6.14	RTC time-stamp sub second register (RTC_TSSSR) . . . . .	1436
42.6.15	RTC calibration register (RTC_CALR) . . . . .	1437
42.6.16	RTC tamper configuration register (RTC_TAMPCR) . . . . .	1438
42.6.17	RTC alarm A sub second register (RTC_ALRMASR) . . . . .	1441
42.6.18	RTC alarm B sub second register (RTC_ALRMBSSR) . . . . .	1442
42.6.19	RTC option register (RTC_OR) . . . . .	1443
42.6.20	RTC backup registers (RTC_BKPxR) . . . . .	1443
42.6.21	RTC register map . . . . .	1444
<b>43</b>	<b>Inter-integrated circuit (I2C) interface . . . . .</b>	<b>1446</b>
43.1	Introduction . . . . .	1446
43.2	I2C main features . . . . .	1446
43.3	I2C implementation . . . . .	1447
43.4	I2C functional description . . . . .	1447
43.4.1	I2C block diagram . . . . .	1448
43.4.2	I2C clock requirements . . . . .	1449
43.4.3	Mode selection . . . . .	1449
43.4.4	I2C initialization . . . . .	1450
43.4.5	Software reset . . . . .	1455
43.4.6	Data transfer . . . . .	1456
43.4.7	I2C slave mode . . . . .	1458
43.4.8	I2C master mode . . . . .	1467
43.4.9	I2C_TIMINGR register configuration examples . . . . .	1479
43.4.10	SMBus specific features . . . . .	1480
43.4.11	SMBus initialization . . . . .	1483
43.4.12	SMBus: I2C_TIMEOUTR register configuration examples . . . . .	1485
43.4.13	SMBus slave mode . . . . .	1486

- 43.4.14 Wakeup from Stop mode on address match ..... 1493
- 43.4.15 Error conditions ..... 1493
- 43.4.16 DMA requests ..... 1495
- 43.4.17 Debug mode ..... 1496
- 43.5 I2C low-power modes ..... 1496
- 43.6 I2C interrupts ..... 1496
- 43.7 I2C registers ..... 1498
  - 43.7.1 Control register 1 (I2C\_CR1) ..... 1498
  - 43.7.2 Control register 2 (I2C\_CR2) ..... 1501
  - 43.7.3 Own address 1 register (I2C\_OAR1) ..... 1504
  - 43.7.4 Own address 2 register (I2C\_OAR2) ..... 1505
  - 43.7.5 Timing register (I2C\_TIMINGR) ..... 1506
  - 43.7.6 Timeout register (I2C\_TIMEOUTR) ..... 1507
  - 43.7.7 Interrupt and status register (I2C\_ISR) ..... 1508
  - 43.7.8 Interrupt clear register (I2C\_ICR) ..... 1510
  - 43.7.9 PEC register (I2C\_PECR) ..... 1511
  - 43.7.10 Receive data register (I2C\_RXDR) ..... 1512
  - 43.7.11 Transmit data register (I2C\_TXDR) ..... 1512
  - 43.7.12 I2C register map ..... 1513

**44 Universal synchronous asynchronous receiver transmitter (USART) ..... 1515**

- 44.1 USART introduction ..... 1515
- 44.2 USART main features ..... 1516
- 44.3 USART extended features ..... 1517
- 44.4 USART implementation ..... 1517
- 44.5 USART functional description ..... 1518
  - 44.5.1 USART block diagram ..... 1518
  - 44.5.2 USART signals ..... 1519
  - 44.5.3 USART character description ..... 1520
  - 44.5.4 USART FIFOs and thresholds ..... 1522
  - 44.5.5 USART transmitter ..... 1522
  - 44.5.6 USART receiver ..... 1526
  - 44.5.7 USART baud rate generation ..... 1533
  - 44.5.8 Tolerance of the USART receiver to clock deviation ..... 1534
  - 44.5.9 USART Auto baud rate detection ..... 1535





- 44.5.10 USART multiprocessor communication ..... 1537
- 44.5.11 USART Modbus communication ..... 1539
- 44.5.12 USART parity control ..... 1540
- 44.5.13 USART LIN (local interconnection network) mode ..... 1541
- 44.5.14 USART synchronous mode ..... 1543
- 44.5.15 USART single-wire Half-duplex communication ..... 1547
- 44.5.16 USART receiver timeout ..... 1547
- 44.5.17 USART Smartcard mode ..... 1548
- 44.5.18 USART IrDA SIR ENDEC block ..... 1552
- 44.5.19 Continuous communication using USART and DMA ..... 1555
- 44.5.20 RS232 Hardware flow control and RS485 Driver Enable ..... 1557
- 44.5.21 USART low-power management ..... 1560
- 44.6 USART interrupts ..... 1563
- 44.7 USART registers ..... 1566
  - 44.7.1 USART control register 1 (USART\_CR1) ..... 1566
  - 44.7.2 USART control register 2 (USART\_CR2) ..... 1571
  - 44.7.3 USART control register 3 (USART\_CR3) ..... 1574
  - 44.7.4 USART baud rate register (USART\_BRR) ..... 1579
  - 44.7.5 USART guard time and prescaler register (USART\_GTPR) ..... 1579
  - 44.7.6 USART receiver timeout register (USART\_RTOR) ..... 1580
  - 44.7.7 USART request register (USART\_RQR) ..... 1581
  - 44.7.8 USART interrupt and status register (USART\_ISR) ..... 1582
  - 44.7.9 USART interrupt flag clear register (USART\_ICR) ..... 1588
  - 44.7.10 USART receive data register (USART\_RDR) ..... 1590
  - 44.7.11 USART transmit data register (USART\_TDR) ..... 1590
  - 44.7.12 USART prescaler register (USART\_PRESC) ..... 1591
  - 44.7.13 USART register map ..... 1592

- 45 Low-power universal asynchronous receiver transmitter (LPUART) ..... 1594**
  - 45.1 LPUART introduction ..... 1594
  - 45.2 LPUART main features ..... 1595
  - 45.3 LPUART functional description ..... 1596
    - 45.3.1 LPUART block diagram ..... 1596
    - 45.3.2 LPUART signals ..... 1597
    - 45.3.3 LPUART character description ..... 1597
    - 45.3.4 LPUART FIFOs and thresholds ..... 1598



45.3.5	LPUART transmitter	1599
45.3.6	LPUART receiver	1602
45.3.7	LPUART baud rate generation	1606
45.3.8	Tolerance of the LPUART receiver to clock deviation	1608
45.3.9	LPUART multiprocessor communication	1608
45.3.10	LPUART parity control	1610
45.3.11	LPUART single-wire Half-duplex communication	1611
45.3.12	Continuous communication using DMA and LPUART	1612
45.3.13	RS232 Hardware flow control and RS485 Driver Enable	1614
45.3.14	LPUART low-power management	1617
45.4	LPUART interrupts	1620
45.5	LPUART registers	1622
45.5.1	Control register 1 (LPUART_CR1)	1622
45.5.2	Control register 2 (LPUART_CR2)	1625
45.5.3	Control register 3 (LPUART_CR3)	1627
45.5.4	Baud rate register (LPUART_BRR)	1630
45.5.5	Request register (LPUART_RQR)	1631
45.5.6	Interrupt and status register (LPUART_ISR)	1631
45.5.7	Interrupt flag clear register (LPUART_ICR)	1635
45.5.8	Receive data register (LPUART_RDR)	1636
45.5.9	Transmit data register (LPUART_TDR)	1637
45.5.10	Prescaler register (LPUART_PRESC)	1637
45.5.11	LPUART register map	1639
<b>46</b>	<b>Serial peripheral interface (SPI)</b>	<b>1640</b>
46.1	Introduction	1640
46.2	SPI main features	1640
46.3	SPI implementation	1640
46.4	SPI functional description	1641
46.4.1	General description	1641
46.4.2	Communications between one master and one slave	1642
46.4.3	Standard multi-slave communication	1644
46.4.4	Multi-master communication	1645
46.4.5	Slave select (NSS) pin management	1646
46.4.6	Communication formats	1647
46.4.7	Configuration of SPI	1649

46.4.8	Procedure for enabling SPI	1650
46.4.9	Data transmission and reception procedures	1650
46.4.10	SPI status flags	1660
46.4.11	SPI error flags	1661
46.4.12	NSS pulse mode	1662
46.4.13	TI mode	1662
46.4.14	CRC calculation	1663
46.5	SPI interrupts	1665
46.6	SPI registers	1666
46.6.1	SPI control register 1 (SPIx_CR1)	1666
46.6.2	SPI control register 2 (SPIx_CR2)	1668
46.6.3	SPI status register (SPIx_SR)	1670
46.6.4	SPI data register (SPIx_DR)	1671
46.6.5	SPI CRC polynomial register (SPIx_CRCPR)	1671
46.6.6	SPI Rx CRC register (SPIx_RXCR)	1672
46.6.7	SPI Tx CRC register (SPIx_TXCR)	1672
46.6.8	SPI register map	1674
<b>47</b>	<b>Serial audio interface (SAI)</b>	<b>1675</b>
47.1	Introduction	1675
47.2	SAI main features	1676
47.3	SAI functional description	1677
47.3.1	SAI block diagram	1677
47.3.2	SAI pins and internal signals	1678
47.3.3	Main SAI modes	1678
47.3.4	SAI synchronization mode	1679
47.3.5	Audio data size	1680
47.3.6	Frame synchronization	1681
47.3.7	Slot configuration	1684
47.3.8	SAI clock generator	1686
47.3.9	Internal FIFOs	1688
47.3.10	PDM Interface	1690
47.3.11	AC'97 link controller	1698
47.3.12	SPDIF output	1700
47.3.13	Specific features	1703
47.3.14	Error flags	1707
47.3.15	Disabling the SAI	1710

47.3.16	SAI DMA interface	1710
47.4	SAI interrupts	1711
47.5	SAI registers	1712
47.5.1	Global configuration register (SAI_GCR)	1712
47.5.2	Configuration register 1 (SAI_ACR1)	1712
47.5.3	Configuration register 1 (SAI_BCR1)	1715
47.5.4	Configuration register 2 (SAI_ACR2)	1718
47.5.5	Configuration register 2 (SAI_BCR2)	1720
47.5.6	Frame configuration register (SAI_AFRCR)	1722
47.5.7	Frame configuration register (SAI_BFRCR)	1723
47.5.8	Slot register (SAI_ASLOTR)	1724
47.5.9	Slot register (SAI_BSLOTR)	1725
47.5.10	Interrupt mask register 2 (SAI_AIM)	1726
47.5.11	Interrupt mask register 2 (SAI_BIM)	1728
47.5.12	Status register (SAI_ASR)	1729
47.5.13	Status register (SAI_BSR)	1731
47.5.14	Clear flag register (SAI_ACLRFR)	1733
47.5.15	Clear flag register (SAI_BCLRFR)	1734
47.5.16	Data register (SAI_ADR)	1735
47.5.17	Data register (SAI_BDR)	1736
47.5.18	PDM control register (SAI_PDMCR)	1736
47.5.19	PDM delay register (SAI_PDMDLY)	1737
47.5.20	SAI register map	1740
<b>48</b>	<b>Secure digital input/output MultiMediaCard interface (SDMMC)</b>	<b>1742</b>
48.1	SDMMC main features	1742
48.2	SDMMC bus topology	1742
48.3	SDMMC operation modes	1744
48.4	SDMMC functional description	1745
48.4.1	SDMMC diagram	1745
48.4.2	SDMMC pins and internal signals	1745
48.4.3	General description	1746
48.4.4	SDMMC adapter	1748
48.4.5	SDMMC AHB slave interface	1769
48.4.6	SDMMC AHB master interface	1770
48.4.7	AHB and SDMMC_CK clock relation	1772

48.5	Card functional description	1772
48.5.1	SD I/O mode	1772
48.5.2	CMD12 send timing	1780
48.5.3	Sleep (CMD5)	1784
48.5.4	Interrupt mode (Wait-IRQ)	1785
48.5.5	Boot operation	1786
48.5.6	Response R1b handling	1789
48.5.7	Reset and card cycle power	1790
48.6	Hardware flow control	1791
48.7	Ultra-high-speed phase I (UHS-I) voltage switch	1792
48.8	SDMMC registers	1795
48.8.1	SDMMC power control register (SDMMC_POWER)	1795
48.8.2	SDMMC clock control register (SDMMC_CLKCR)	1796
48.8.3	SDMMC argument register (SDMMC_ARGR)	1798
48.8.4	SDMMC command register (SDMMC_CMDR)	1798
48.8.5	SDMMC command response register (SDMMC_RESPCMDR)	1800
48.8.6	SDMMC response x register (SDMMC_RESPxR)	1801
48.8.7	SDMMC data timer register (SDMMC_DTIMER)	1801
48.8.8	SDMMC data length register (SDMMC_DLENR)	1802
48.8.9	SDMMC data control register (SDMMC_DCTRL)	1803
48.8.10	SDMMC data counter register (SDMMC_DCNTR)	1804
48.8.11	SDMMC status register (SDMMC_STAR)	1805
48.8.12	SDMMC interrupt clear register (SDMMC_ICR)	1807
48.8.13	SDMMC mask register (SDMMC_MASKR)	1810
48.8.14	SDMMC acknowledgment timer register (SDMMC_ACKTIMER)	1812
48.8.15	SDMMC data FIFO register (SDMMC_FIFOR)	1813
48.8.16	SDMMC DMA control register (SDMMC_IDMACTRLR)	1813
48.8.17	SDMMC IDMA buffer size register (SDMMC_IDMABSIZER)	1814
48.8.18	SDMMC IDMA buffer 0 base address register (SDMMC_IDMABASE0R)	1815
48.8.19	SDMMC IDMA buffer 1 base address register (SDMMC_IDMABASE1R)	1815
48.8.20	SDMMC register map	1816
<b>49</b>	<b>Controller area network (bxCAN)</b>	<b>1819</b>
49.1	Introduction	1819
49.2	bxCAN main features	1819

49.3	bxCAN general description . . . . .	1820
49.3.1	CAN 2.0B active core . . . . .	1820
49.3.2	Control, status and configuration registers . . . . .	1820
49.3.3	Tx mailboxes . . . . .	1820
49.3.4	Acceptance filters . . . . .	1821
49.4	bxCAN operating modes . . . . .	1822
49.4.1	Initialization mode . . . . .	1822
49.4.2	Normal mode . . . . .	1822
49.4.3	Sleep mode (low-power) . . . . .	1823
49.5	Test mode . . . . .	1824
49.5.1	Silent mode . . . . .	1824
49.5.2	Loop back mode . . . . .	1824
49.5.3	Loop back combined with silent mode . . . . .	1825
49.6	Behavior in debug mode . . . . .	1825
49.7	bxCAN functional description . . . . .	1825
49.7.1	Transmission handling . . . . .	1825
49.7.2	Time triggered communication mode . . . . .	1827
49.7.3	Reception handling . . . . .	1827
49.7.4	Identifier filtering . . . . .	1829
49.7.5	Message storage . . . . .	1833
49.7.6	Error management . . . . .	1835
49.7.7	Bit timing . . . . .	1835
49.8	bxCAN interrupts . . . . .	1838
49.9	CAN registers . . . . .	1839
49.9.1	Register access protection . . . . .	1839
49.9.2	CAN control and status registers . . . . .	1839
49.9.3	CAN mailbox registers . . . . .	1849
49.9.4	CAN filter registers . . . . .	1856
49.9.5	bxCAN register map . . . . .	1860
<b>50</b>	<b>USB on-the-go full-speed (OTG_FS) . . . . .</b>	<b>1864</b>
50.1	Introduction . . . . .	1864
50.2	OTG main features . . . . .	1865
50.2.1	General features . . . . .	1865
50.2.2	Host-mode features . . . . .	1866
50.2.3	Peripheral-mode features . . . . .	1866

50.3	OTG implementation	1867
50.4	OTG functional description	1868
50.4.1	OTG block diagram	1868
50.4.2	USB OTG pin and internal signals	1868
50.4.3	OTG core	1869
50.4.4	Full-speed OTG PHY	1869
50.5	OTG dual role device (DRD)	1870
50.5.1	ID line detection	1870
50.5.2	HNP dual role device	1871
50.5.3	SRP dual role device	1871
50.6	USB peripheral	1871
50.6.1	SRP-capable peripheral	1872
50.6.2	Peripheral states	1872
50.6.3	Peripheral endpoints	1873
50.7	USB host	1875
50.7.1	SRP-capable host	1876
50.7.2	USB host states	1876
50.7.3	Host channels	1878
50.7.4	Host scheduler	1879
50.8	SOF trigger	1880
50.8.1	Host SOFs	1880
50.8.2	Peripheral SOFs	1880
50.9	OTG low-power modes	1881
50.10	Dynamic update of the OTG_HFIR register	1882
50.11	USB data FIFOs	1882
50.11.1	Peripheral FIFO architecture	1883
50.11.2	Host FIFO architecture	1884
50.11.3	FIFO RAM allocation	1885
50.12	OTG_FS system performance	1887
50.13	OTG_FS interrupts	1887
50.14	OTG_FS control and status registers	1889
50.14.1	CSR memory map	1889
50.15	OTG_FS registers	1893
50.15.1	OTG control and status register (OTG_GOTGCTL)	1894
50.15.2	OTG interrupt register (OTG_GOTGINT)	1897
50.15.3	OTG AHB configuration register (OTG_GAHBCFG)	1898

50.15.4	OTG USB configuration register (OTG_GUSBCFG) . . . . .	1899
50.15.5	OTG reset register (OTG_GRSTCTL) . . . . .	1901
50.15.6	OTG core interrupt register (OTG_GINTSTS) . . . . .	1904
50.15.7	OTG interrupt mask register (OTG_GINTMSK) . . . . .	1908
50.15.8	OTG receive status debug read/OTG status read and pop registers (OTG_GRXSTSR/OTG_GRXSTSP) . . . . .	1911
50.15.9	OTG receive FIFO size register (OTG_GRXFSIZ) . . . . .	1913
50.15.10	OTG host non-periodic transmit FIFO size register (OTG_HNPTXFSIZ)/Endpoint 0 Transmit FIFO size (OTG_DIEPTXF0) . . . . .	1913
50.15.11	OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS) . . . . .	1914
50.15.12	OTG general core configuration register (OTG_GCCFG) . . . . .	1915
50.15.13	OTG core ID register (OTG_CID) . . . . .	1917
50.15.14	OTG core LPM configuration register (OTG_GLPMCFG) . . . . .	1917
50.15.15	OTG power down register (OTG_GPWRDN) . . . . .	1921
50.15.16	OTG ADP timer, control and status register (OTG_GADPCTL) . . . . .	1921
50.15.17	OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ) . . . . .	1923
50.15.18	OTG device IN endpoint transmit FIFO size register (OTG_DIEPTxFx) (x = 1..5, where x is the FIFO number) . . . . .	1924
50.15.19	Host-mode registers . . . . .	1924
50.15.20	OTG host configuration register (OTG_HCFG) . . . . .	1924
50.15.21	OTG host frame interval register (OTG_HFIR) . . . . .	1925
50.15.22	OTG host frame number/frame time remaining register (OTG_HFNUM) . . . . .	1926
50.15.23	OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS) . . . . .	1927
50.15.24	OTG host all channels interrupt register (OTG_HAINT) . . . . .	1928
50.15.25	OTG host all channels interrupt mask register (OTG_HAINTMSK) . . . . .	1928
50.15.26	OTG host port control and status register (OTG_HPRT) . . . . .	1929
50.15.27	OTG host channel x characteristics register (OTG_HCCHARx) (x = 0..11, where x = Channel number) . . . . .	1931
50.15.28	OTG host channel x interrupt register (OTG_HCINTx) (x = 0..11, where x = Channel number) . . . . .	1932
50.15.29	OTG host channel x interrupt mask register (OTG_HCINTMSKx) (x = 0..11, where x = Channel number) . . . . .	1933



50.15.30 OTG host channel x transfer size register (OTG_HCTSIZx) (x = 0..11, where x = Channel number) .....	1934
50.15.31 Device-mode registers .....	1935
50.15.32 OTG device configuration register (OTG_DCFG) .....	1935
50.15.33 OTG device control register (OTG_DCTL) .....	1936
50.15.34 OTG device status register (OTG_DSTS) .....	1939
50.15.35 OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK) .....	1940
50.15.36 OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK) .....	1941
50.15.37 OTG device all endpoints interrupt register (OTG_DAINTR) .....	1942
50.15.38 OTG all endpoints interrupt mask register (OTG_DAINTRMSK) .....	1943
50.15.39 OTG device V <sub>BUS</sub> discharge time register (OTG_DVBUSDIS) .....	1943
50.15.40 OTG device V <sub>BUS</sub> pulsing time register (OTG_DVBUSPULSE) .....	1944
50.15.41 OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK) .....	1944
50.15.42 OTG device control IN endpoint 0 control register (OTG_DIEPCTL0) .....	1945
50.15.43 OTG device IN endpoint x control register (OTG_DIEPCTLx) (x = 1..5, where x = endpoint number) .....	1946
50.15.44 OTG device IN endpoint x interrupt register (OTG_DIEPINTx) (x = 0..5, where x = Endpoint number) .....	1949
50.15.45 OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0) .....	1950
50.15.46 OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx) (x = 0..5, where x = endpoint number) .....	1951
50.15.47 OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx) (x = 1..5, where x = endpoint number) .....	1952
50.15.48 OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0) .....	1953
50.15.49 OTG device OUT endpoint x interrupt register (OTG_DOEPINTx) (x = 0..5, where x = Endpoint number) .....	1954
50.15.50 OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0) .....	1956
50.15.51 OTG device OUT endpoint x control register (OTG_DOEPCTLx) (x = 1..5, where x = endpoint number) .....	1957
50.15.52 OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx) (x = 1..5, where x = Endpoint number) .....	1959

50.15.53	OTG power and clock gating control register (OTG_PCGCCTL) . . .	1960
50.15.54	OTG_FS register map . . . . .	1961
50.16	OTG_FS programming model . . . . .	1969
50.16.1	Core initialization . . . . .	1969
50.16.2	Host initialization . . . . .	1969
50.16.3	Device initialization . . . . .	1970
50.16.4	Host programming model . . . . .	1971
50.16.5	Device programming model . . . . .	1992
50.16.6	Worst case response time . . . . .	2011
50.16.7	OTG programming model . . . . .	2013
<b>51</b>	<b>Debug support (DBG) . . . . .</b>	<b>2019</b>
51.1	Overview . . . . .	2019
51.2	Reference Arm® documentation . . . . .	2020
51.3	SWJ debug port (serial wire and JTAG) . . . . .	2020
51.3.1	Mechanism to select the JTAG-DP or the SW-DP . . . . .	2021
51.4	Pinout and debug port pins . . . . .	2021
51.4.1	SWJ debug port pins . . . . .	2022
51.4.2	Flexible SWJ-DP pin assignment . . . . .	2022
51.4.3	Internal pull-up and pull-down on JTAG pins . . . . .	2023
51.4.4	Using serial wire and releasing the unused debug pins as GPIOs . .	2024
51.5	STM32L4Rxxx and STM32L4Sxxx JTAG TAP connection . . . . .	2024
51.6	ID codes and locking mechanism . . . . .	2025
51.6.1	MCU device ID code . . . . .	2026
51.6.2	Boundary scan TAP . . . . .	2026
51.6.3	Cortex®-M4 TAP . . . . .	2026
51.6.4	Cortex®-M4 JEDEC-106 ID code . . . . .	2026
51.7	JTAG debug port . . . . .	2027
51.8	SW debug port . . . . .	2028
51.8.1	SW protocol introduction . . . . .	2028
51.8.2	SW protocol sequence . . . . .	2028
51.8.3	SW-DP state machine (reset, idle states, ID code) . . . . .	2029
51.8.4	DP and AP read/write accesses . . . . .	2030
51.8.5	SW-DP registers . . . . .	2030
51.8.6	SW-AP registers . . . . .	2032

51.9	AHB-AP (AHB access port) - valid for both JTAG-DP and SW-DP	2032
51.10	Core debug	2033
51.11	Capability of the debugger host to connect under system reset	2033
51.12	FPB (Flash patch breakpoint)	2034
51.13	DWT (data watchpoint trigger)	2034
51.14	ITM (instrumentation trace macrocell)	2035
51.14.1	General description	2035
51.14.2	Time stamp packets, synchronization and overflow packets	2035
51.15	ETM (Embedded trace macrocell)	2037
51.15.1	General description	2037
51.15.2	Signal protocol, packet types	2037
51.15.3	Main ETM registers	2037
51.15.4	Configuration example	2038
51.16	MCU debug component (DBGMCU)	2038
51.16.1	Debug support for low-power modes	2038
51.16.2	Debug support for timers, RTC, watchdog, bxCAN and I <sup>2</sup> C	2039
51.16.3	Debug MCU configuration register (DBGMCU_CR)	2039
51.16.4	Debug MCU APB1 freeze register1(DBGMCU_APB1FZR1)	2040
51.16.5	Debug MCU APB1 freeze register 2 (DBGMCU_APB1FZR2)	2042
51.16.6	Debug MCU APB2 freeze register (DBGMCU_APB2FZR)	2042
51.17	TPIU (trace port interface unit)	2044
51.17.1	Introduction	2044
51.17.2	TRACE pin assignment	2044
51.17.3	TPUI formatter	2046
51.17.4	TPUI frame synchronization packets	2047
51.17.5	Transmission of the synchronization frame packet	2047
51.17.6	Synchronous mode	2047
51.17.7	Asynchronous mode	2048
51.17.8	TRACECLKIN connection inside the STM32L4Rxxx and STM32L4Sxxx	2048
51.17.9	TPIU registers	2049
51.17.10	Example of configuration	2050
51.18	DBG register map	2051
<b>52</b>	<b>Device electronic signature</b>	<b>2052</b>

52.1	Unique device ID register (96 bits) .....	2052
52.2	Flash size data register .....	2053
52.3	Package data register .....	2054
<b>53</b>	<b>Revision history .....</b>	<b>2055</b>

## List of tables

Table 1.	STM32L4Rxxx and STM32L4Sxxx memory map and peripheral register boundary addresses	83
Table 2.	SRAM2 organization	90
Table 3.	Boot modes	93
Table 4.	Memory mapping versus boot mode/physical remap	94
Table 5.	Flash module - 2 Mbytes dual-bank organization (64 bits read width)	99
Table 6.	Flash module - 2 Mbytes single-bank organization (128 bits read width)	99
Table 7.	Flash module - 1 Mbyte dual-bank organization (64 bits read width)	100
Table 8.	Flash module - 1 Mbyte single-bank organization (128 bits read width)	101
Table 9.	Number of wait states according to CPU clock (HCLK) frequency	103
Table 10.	Option byte format	114
Table 11.	Option byte organization	114
Table 12.	Flash memory read protection status	124
Table 13.	Access status versus protection level and execution modes	126
Table 14.	PCROP protection	128
Table 15.	WRP protection	130
Table 16.	Flash interrupt request	130
Table 17.	Flash interface - register map and reset values	148
Table 18.	Segment accesses according to the Firewall state	153
Table 19.	Segment granularity and area ranges	154
Table 20.	Firewall register map and reset values	162
Table 21.	Range 1 boost mode configuration	171
Table 22.	PVM features	174
Table 23.	Low-power mode summary	178
Table 24.	Functionalities depending on the working mode	179
Table 25.	Low-power run	183
Table 26.	Sleep	184
Table 27.	Low-power sleep	186
Table 28.	Stop 0 mode	188
Table 29.	Stop 1 mode	189
Table 30.	Stop 2 mode	191
Table 31.	Standby mode	193
Table 32.	Shutdown mode	195
Table 33.	PWR register map and reset values	214
Table 34.	Clock source frequency	228
Table 35.	RCC register map and reset values	291
Table 36.	Effect of low-power modes on CRS	300
Table 37.	Interrupt control bits	300
Table 38.	CRS register map and reset values	306
Table 39.	Port bit configuration table	309
Table 40.	GPIO register map and reset values	323
Table 41.	BOOSTEN and ANASWVDD set/reset	328
Table 42.	SYSCFG register map and reset values	338
Table 43.	STM32L4Rxxx and STM32L4Sxxx peripherals interconnect matrix	340
Table 44.	DMA1 and DMA2 implementation	349
Table 45.	DMA internal input/output signals	351
Table 46.	Programmable data width and endian behavior (when PINC = MINC = 1)	356
Table 47.	DMA interrupt requests	358

Table 48.	DMA register map and reset values . . . . .	366
Table 49.	DMAMUX instantiation . . . . .	370
Table 50.	DMAMUX: assignment of multiplexer inputs to resources . . . . .	370
Table 51.	DMAMUX: assignment of trigger inputs to resources . . . . .	371
Table 52.	DMAMUX: assignment of synchronization inputs to resources . . . . .	372
Table 53.	DMAMUX signals . . . . .	374
Table 54.	DMAMUX interrupts . . . . .	378
Table 55.	DMAMUX register map and reset values . . . . .	383
Table 56.	Supported color mode in input . . . . .	388
Table 57.	Data order in memory . . . . .	389
Table 58.	Alpha mode configuration . . . . .	390
Table 59.	Supported CLUT color mode . . . . .	391
Table 60.	CLUT data order in system memory . . . . .	391
Table 61.	Supported color mode in output . . . . .	392
Table 62.	Data order in memory . . . . .	392
Table 63.	Standard data order in memory . . . . .	393
Table 64.	Output FIFO byte reordering steps . . . . .	394
Table 65.	DMA2D interrupt requests . . . . .	400
Table 66.	DMA2D register map and reset values . . . . .	419
Table 67.	Graphic MMU interrupt requests . . . . .	428
Table 68.	Graphic MMU register map and reset values . . . . .	435
Table 69.	STM32L4Rxxx and STM32L4Sxxx vector table . . . . .	437
Table 70.	EXTI lines connections . . . . .	444
Table 71.	Extended interrupt/event controller register map and reset values . . . . .	453
Table 72.	CRC internal input/output signals . . . . .	455
Table 73.	CRC register map and reset values . . . . .	459
Table 74.	NOR/PSRAM bank selection . . . . .	464
Table 75.	NOR/PSRAM External memory address . . . . .	464
Table 76.	NAND memory mapping and timing registers . . . . .	465
Table 77.	NAND bank selection . . . . .	465
Table 78.	Programmable NOR/PSRAM access parameters . . . . .	467
Table 79.	Non-multiplexed I/O NOR Flash memory . . . . .	467
Table 80.	16-bit multiplexed I/O NOR Flash memory . . . . .	468
Table 81.	Non-multiplexed I/Os PSRAM/SRAM . . . . .	468
Table 82.	16-Bit multiplexed I/O PSRAM . . . . .	468
Table 83.	NOR Flash/PSRAM: example of supported memories and transactions . . . . .	469
Table 84.	FSMC_BCRx bit fields . . . . .	472
Table 85.	FSMC_BTRx bit fields . . . . .	473
Table 86.	FSMC_BCRx bit fields . . . . .	475
Table 87.	FSMC_BTRx bit fields . . . . .	475
Table 88.	FSMC_BWTRx bit fields . . . . .	476
Table 89.	FSMC_BCRx bit fields . . . . .	478
Table 90.	FSMC_BTRx bit fields . . . . .	478
Table 91.	FSMC_BWTRx bit fields . . . . .	479
Table 92.	FSMC_BCRx bit fields . . . . .	480
Table 93.	FSMC_BTRx bit fields . . . . .	481
Table 94.	FSMC_BWTRx bit fields . . . . .	481
Table 95.	FSMC_BCRx bit fields . . . . .	483
Table 96.	FSMC_BTRx bit fields . . . . .	484
Table 97.	FSMC_BWTRx bit fields . . . . .	484
Table 98.	FSMC_BCRx bit fields . . . . .	486
Table 99.	FSMC_BTRx bit fields . . . . .	487

Table 100.	FSMC_BCRx bit fields	492
Table 101.	FSMC_BTRx bit fields	493
Table 102.	FSMC_BCRx bit fields	494
Table 103.	FSMC_BTRx bit fields	495
Table 104.	Programmable NAND Flash access parameters	503
Table 105.	8-bit NAND Flash	504
Table 106.	16-bit NAND Flash	505
Table 107.	Supported memories and transactions	506
Table 108.	ECC result relevant bits	515
Table 109.	FSMC register map	516
Table 110.	Command/address phase description	528
Table 111.	OCTOSPI interrupt requests	544
Table 112.	OCTOSPI register map	565
Table 113.	OCTOSPI IO Manager register map and reset values	572
Table 114.	ADC internal input/output signals	578
Table 115.	ADC input/output pins	578
Table 116.	Configuring the trigger polarity for regular external triggers	595
Table 117.	Configuring the trigger polarity for injected external triggers	595
Table 118.	ADC1 - External triggers for regular channels	596
Table 119.	ADC1 - External trigger for injected channels	596
Table 120.	TSAR timings depending on resolution	608
Table 121.	Offset computation versus data resolution	611
Table 122.	Analog watchdog channel selection	622
Table 123.	Analog watchdog 1 comparison	623
Table 124.	Analog watchdog 2 and 3 comparison	623
Table 125.	Maximum output results versus N and M (gray cells indicate truncation)	627
Table 126.	ADC interrupts	635
Table 127.	ADC register map and reset values	664
Table 128.	ADC register map and reset values (master and slave ADC common registers) offset = 0x300	667
Table 129.	DAC implementation	669
Table 130.	DAC input/output pins	671
Table 131.	DAC trigger selection	674
Table 132.	Sample and refresh timings	678
Table 133.	Channel output modes summary	680
Table 134.	Effect of low-power modes on DAC	687
Table 135.	DAC interrupts	687
Table 136.	DAC register map and reset values	702
Table 137.	DCMI external signals	706
Table 138.	Positioning of captured data bytes in 32-bit words (8-bit width)	707
Table 139.	Positioning of captured data bytes in 32-bit words (10-bit width)	707
Table 140.	Positioning of captured data bytes in 32-bit words (12-bit width)	707
Table 141.	Positioning of captured data bytes in 32-bit words (14-bit width)	708
Table 142.	Data storage in monochrome progressive video format	713
Table 143.	Data storage in RGB progressive video format	714
Table 144.	Data storage in YCbCr progressive video format	714
Table 145.	Data storage in YCbCr progressive video format - Y extraction mode	715
Table 146.	DCMI interrupts	715
Table 147.	DCMI register map and reset values	728
Table 148.	VREF buffer modes	729
Table 149.	VREFBUF register map and reset values	731
Table 150.	COMP1 input plus assignment	733

Table 151.	COMP1 input minus assignment . . . . .	733
Table 152.	COMP2 input plus assignment . . . . .	734
Table 153.	COMP2 input minus assignment . . . . .	734
Table 154.	Comparator behavior in the low power modes . . . . .	737
Table 155.	Interrupt control bits . . . . .	738
Table 156.	COMP register map and reset values . . . . .	743
Table 157.	Operational amplifier possible connections . . . . .	745
Table 158.	Operating modes and calibration . . . . .	750
Table 159.	Effect of low-power modes on the OPAMP . . . . .	751
Table 160.	OPAMP register map and reset values . . . . .	756
Table 161.	DFSDM1 implementation . . . . .	759
Table 162.	DFSDM external pins . . . . .	761
Table 163.	DFSDM internal signals . . . . .	761
Table 164.	DFSDM triggers connection . . . . .	761
Table 165.	DFSDM break connection . . . . .	762
Table 166.	Filter maximum output resolution (peak data values from filter output) for some FOSR values . . . . .	776
Table 167.	Integrator maximum output resolution (peak data values from integrator output) for some IOSR values and FOSR = 256 and Sinc3 filter type (largest data) . . . . .	777
Table 168.	DFSDM interrupt requests . . . . .	785
Table 169.	DFSDM register map and reset values . . . . .	806
Table 170.	LTDC pins and signal interface . . . . .	817
Table 171.	Clock domain for each register . . . . .	818
Table 172.	Pixel data mapping versus color format . . . . .	823
Table 173.	LTDC interrupt requests . . . . .	827
Table 174.	LTDC register map and reset values . . . . .	848
Table 175.	Location of color components in the LTDC interface . . . . .	856
Table 176.	Multiplicity of the payload size in pixels for each data type . . . . .	857
Table 177.	Contention detection timeout counters configuration . . . . .	869
Table 178.	List of events of different categories of the PRESP_TO counter . . . . .	870
Table 179.	PRESP_TO counter configuration . . . . .	873
Table 180.	Frame requirement configuration registers . . . . .	885
Table 181.	RGB components . . . . .	887
Table 182.	Slew-rate and delay tuning . . . . .	889
Table 183.	Custom lane configuration . . . . .	890
Table 184.	Custom timing parameters . . . . .	890
Table 185.	DSI wrapper interrupt requests . . . . .	894
Table 186.	Error causes and recovery . . . . .	895
Table 187.	DSIHOST register map and reset values . . . . .	967
Table 188.	Acquisition sequence summary . . . . .	975
Table 189.	Spread spectrum deviation versus AHB clock frequency . . . . .	977
Table 190.	I/O state depending on its mode and IODEF bit value . . . . .	978
Table 191.	Effect of low-power modes on TSC . . . . .	980
Table 192.	Interrupt control bits . . . . .	980
Table 193.	TSC register map and reset values . . . . .	989
Table 194.	RNG internal input/output signals . . . . .	992
Table 195.	RNG interrupt requests . . . . .	999
Table 196.	RNG register map and reset map . . . . .	1004
Table 197.	AES internal input/output signals . . . . .	1006
Table 198.	CTR mode initialization vector definition . . . . .	1025
Table 199.	GCM last block definition . . . . .	1027
Table 200.	GCM mode IVI bitfield initialization . . . . .	1028



Table 201.	Initialization of AES_IVRx registers in CCM mode . . . . .	1036
Table 202.	Key endianness in AES_KEYRx registers (128- or 256-bit key length) . . . . .	1041
Table 203.	DMA channel configuration for memory-to-AES data transfer . . . . .	1042
Table 204.	DMA channel configuration for AES-to-memory data transfer . . . . .	1043
Table 205.	AES interrupt requests . . . . .	1045
Table 206.	Processing latency (in clock cycle) for ECB, CBC and CTR. . . . .	1045
Table 207.	Processing latency for GCM and CCM (in clock cycle) . . . . .	1045
Table 208.	AES register map and reset values . . . . .	1057
Table 209.	HASH internal input/output signals . . . . .	1060
Table 210.	Hash processor outputs . . . . .	1064
Table 211.	HASH interrupt requests. . . . .	1070
Table 212.	Processing time (in clock cycle) . . . . .	1070
Table 213.	HASH register map and reset values . . . . .	1082
Table 214.	Behavior of timer outputs versus BRK/BRK2 inputs . . . . .	1124
Table 215.	Break protection disarming conditions . . . . .	1126
Table 216.	Counting direction versus encoder signals . . . . .	1132
Table 217.	TIMx internal trigger connection . . . . .	1149
Table 218.	Output control bits for complementary OCx and OCxN channels with break feature. . . . .	1163
Table 219.	TIM1 register map and reset values . . . . .	1183
Table 220.	TIM8 register map and reset values . . . . .	1186
Table 221.	Counting direction versus encoder signals . . . . .	1223
Table 222.	TIMx internal trigger connection . . . . .	1239
Table 223.	Output control bit for standard OCx channels. . . . .	1250
Table 224.	TIM2/TIM3/TIM4/TIM5 register map and reset values . . . . .	1257
Table 225.	Break protection disarming conditions . . . . .	1288
Table 226.	TIMx Internal trigger connection . . . . .	1303
Table 227.	Output control bits for complementary OCx and OCxN channels with break feature (TIM15). . . . .	1313
Table 228.	TIM15 register map and reset values . . . . .	1322
Table 229.	Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17) . . . . .	1334
Table 230.	TIM16/TIM17 register map and reset values . . . . .	1345
Table 231.	TIM6/TIM7 register map and reset values . . . . .	1359
Table 232.	STM32L4Rxxx and STM32L4Sxxx LPTIM features . . . . .	1360
Table 233.	LPTIM1 external trigger connection . . . . .	1361
Table 234.	LPTIM2 external trigger connection . . . . .	1362
Table 235.	Prescaler division ratios . . . . .	1363
Table 236.	Encoder counting scenarios . . . . .	1370
Table 237.	Effect of low-power modes on the LPTIM. . . . .	1371
Table 238.	Interrupt events. . . . .	1372
Table 239.	LPTIM register map and reset values . . . . .	1382
Table 240.	IWDG register map and reset values . . . . .	1393
Table 241.	WWDG register map and reset values . . . . .	1400
Table 242.	RTC pin PC13 configuration . . . . .	1404
Table 243.	RTC_OUT mapping . . . . .	1405
Table 244.	RTC functions over modes . . . . .	1406
Table 245.	Effect of low-power modes on RTC . . . . .	1418
Table 246.	Interrupt control bits . . . . .	1419
Table 247.	RTC register map and reset values . . . . .	1444
Table 248.	I2C implementation. . . . .	1447
Table 249.	Comparison of analog vs. digital filters . . . . .	1451
Table 250.	I2C-SMBUS specification data setup and hold times . . . . .	1454

Table 251.	I2C configuration . . . . .	1458
Table 252.	I2C-SMBUS specification clock timings . . . . .	1469
Table 253.	Examples of timing settings for fI2CCLK = 8 MHz . . . . .	1479
Table 254.	Examples of timings settings for fI2CCLK = 16 MHz . . . . .	1479
Table 255.	Examples of timings settings for fI2CCLK = 48 MHz . . . . .	1480
Table 256.	SMBus timeout specifications . . . . .	1482
Table 257.	SMBUS with PEC configuration . . . . .	1484
Table 258.	Examples of TIMEOUTA settings for various I2CCLK frequencies (max $t_{\text{TIMEOUT}} = 25 \text{ ms}$ ) . . . . .	1485
Table 259.	Examples of TIMEOUTB settings for various I2CCLK frequencies . . . . .	1485
Table 260.	Examples of TIMEOUTA settings for various I2CCLK frequencies (max $t_{\text{IDLE}} = 50 \mu\text{s}$ ) . . . . .	1485
Table 261.	Effect of low-power modes on the I2C . . . . .	1496
Table 262.	I2C Interrupt requests . . . . .	1496
Table 263.	I2C register map and reset values . . . . .	1513
Table 264.	USART/LPUART features . . . . .	1517
Table 265.	Noise detection from sampled data . . . . .	1531
Table 266.	Tolerance of the USART receiver when BRR [3:0] = 0000. . . . .	1535
Table 267.	Tolerance of the USART receiver when BRR[3:0] is different from 0000 . . . . .	1535
Table 268.	USART frame formats . . . . .	1540
Table 269.	USART interrupt requests. . . . .	1563
Table 270.	USART register map and reset values . . . . .	1592
Table 271.	Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32,768 KHz . . . . .	1607
Table 272.	Error calculation for programmed baud rates at fCK = 100 MHz . . . . .	1607
Table 273.	Tolerance of the LPUART receiver . . . . .	1608
Table 275.	LPUART interrupt requests. . . . .	1620
Table 276.	LPUART register map and reset values . . . . .	1639
Table 277.	SPI interrupt requests . . . . .	1665
Table 278.	SPI register map and reset values . . . . .	1674
Table 279.	SAI internal input/output signals . . . . .	1678
Table 280.	SAI input/output pins. . . . .	1678
Table 281.	External synchronization selection . . . . .	1680
Table 282.	Clock generator programming examples . . . . .	1688
Table 283.	TDM settings. . . . .	1695
Table 284.	Allowed TDM frame configuration. . . . .	1697
Table 285.	SOPD pattern . . . . .	1701
Table 286.	Parity bit calculation . . . . .	1701
Table 287.	Audio sampling frequency versus symbol rates . . . . .	1702
Table 288.	SAI interrupt sources . . . . .	1711
Table 289.	SAI register map and reset values . . . . .	1740
Table 290.	SDMMC operation modes SD & SDIO . . . . .	1744
Table 291.	SDMMC operation modes eMMC . . . . .	1745
Table 292.	SDMMC internal input/output signals . . . . .	1746
Table 293.	SDMMC pins. . . . .	1746
Table 294.	SDMMC Command and data phase selection . . . . .	1747
Table 295.	Command token format . . . . .	1753
Table 296.	Short response with CRC token format . . . . .	1754
Table 297.	Short response without CRC token format . . . . .	1754
Table 298.	Long response with CRC token format. . . . .	1754
Table 299.	Specific Commands overview. . . . .	1755
Table 300.	Command path status flags . . . . .	1756
Table 301.	Command path error handling . . . . .	1756

Table 302.	Data token format . . . . .	1764
Table 303.	Data path status flags and clear bits . . . . .	1764
Table 304.	Data path error handling . . . . .	1766
Table 305.	Data FIFO access . . . . .	1767
Table 306.	Transmit FIFO status flags . . . . .	1768
Table 307.	Receive FIFO status flags . . . . .	1769
Table 308.	AHB and SDMMC_CK clock frequency relation . . . . .	1772
Table 309.	SDIO special operation control . . . . .	1772
Table 310.	4-bit mode Start, interrupt, and CRC-status Signaling detection . . . . .	1776
Table 311.	CMD12 use cases . . . . .	1781
Table 312.	Response type and SDMMC_RESPxR registers . . . . .	1801
Table 313.	SDMMC register map . . . . .	1816
Table 314.	Transmit mailbox mapping . . . . .	1834
Table 315.	Receive mailbox mapping . . . . .	1834
Table 316.	bxCAN register map and reset values . . . . .	1860
Table 317.	OTG_FS speeds supported . . . . .	1864
Table 318.	OTG implementation . . . . .	1867
Table 319.	OTG_FS input/output pins . . . . .	1868
Table 320.	OTG_FS input/output signals . . . . .	1869
Table 321.	Compatibility of STM32 low power modes with the OTG . . . . .	1881
Table 322.	Core global control and status registers (CSRs) . . . . .	1889
Table 323.	Host-mode control and status registers (CSRs) . . . . .	1890
Table 324.	Device-mode control and status registers . . . . .	1891
Table 325.	Data FIFO (DFIFO) access register map . . . . .	1893
Table 326.	Power and clock gating control and status registers . . . . .	1893
Table 327.	TRDT values (FS) . . . . .	1901
Table 328.	Minimum duration for soft disconnect . . . . .	1938
Table 329.	OTG_FS register map and reset values . . . . .	1961
Table 330.	SWJ debug port pins . . . . .	2022
Table 331.	Flexible SWJ-DP pin assignment . . . . .	2022
Table 332.	JTAG debug port data registers . . . . .	2027
Table 333.	32-bit debug port registers addressed through the shifted value A[3:2] . . . . .	2028
Table 334.	Packet request (8-bits) . . . . .	2029
Table 335.	ACK response (3 bits) . . . . .	2029
Table 336.	DATA transfer (33 bits) . . . . .	2029
Table 337.	SW-DP registers . . . . .	2030
Table 338.	Cortex®-M4 AHB-AP registers . . . . .	2032
Table 339.	Core debug registers . . . . .	2033
Table 340.	Main ITM registers . . . . .	2035
Table 341.	Main ETM registers . . . . .	2037
Table 342.	Asynchronous TRACE pin assignment . . . . .	2044
Table 343.	Synchronous TRACE pin assignment . . . . .	2045
Table 344.	Flexible TRACE pin assignment . . . . .	2045
Table 345.	Important TPIU registers . . . . .	2049
Table 346.	DBG register map and reset values . . . . .	2051
Table 347.	Document revision history . . . . .	2055

## List of figures

Figure 1.	System architecture	78
Figure 2.	Memory map	82
Figure 3.	Sequential 16-bit instructions execution (64-bit read data width)	105
Figure 4.	Changing the Read protection (RDP) level	126
Figure 5.	STM32L4Rxxx and STM32L4Sxxx firewall connection schematics	151
Figure 6.	Firewall functional states	155
Figure 7.	STM32L4S5xx/R5xx and STM32L4S7xx/L4R7xx power supply overview	164
Figure 8.	STM32L4S9xx/L4R9xx power supply overview	165
Figure 9.	Internal main regulator overview	169
Figure 10.	Brown-out reset waveform	173
Figure 11.	PVD thresholds	174
Figure 12.	Low-power modes possible transitions	177
Figure 13.	Simplified diagram of the reset circuit	217
Figure 14.	Clock tree	221
Figure 15.	DSI clock tree	222
Figure 16.	HSE/ LSE clock sources	223
Figure 17.	Frequency measurement with TIM15 in capture mode	231
Figure 18.	Frequency measurement with TIM16 in capture mode	232
Figure 19.	Frequency measurement with TIM17 in capture mode	232
Figure 20.	CRS block diagram	297
Figure 21.	CRS counter behavior	298
Figure 22.	Basic structure of an I/O port bit	308
Figure 23.	Basic structure of a 5-Volt tolerant I/O port bit	308
Figure 24.	Input floating/pull up/pull down configurations	313
Figure 25.	Output configuration	314
Figure 26.	Alternate function configuration	314
Figure 27.	High impedance-analog configuration	315
Figure 28.	DMA block diagram	350
Figure 29.	DMAMUX block diagram	373
Figure 30.	Synchronization mode of the DMAMUX request line multiplexer channel	376
Figure 31.	Event generation of the DMA request line multiplexer channel	376
Figure 32.	DMA2D block diagram	387
Figure 33.	Intel 8080 16-bit mode (RGB565)	393
Figure 34.	Intel 8080 18/24-bit mode (RGB888)	394
Figure 35.	Chrom-GRC™ block diagram	422
Figure 36.	Virtual buffer	423
Figure 37.	Virtual buffer and physical buffer memory map	424
Figure 38.	MMU block diagram	425
Figure 39.	Block validation/comparator implementation	426
Figure 40.	Configurable interrupt/event block diagram	442
Figure 41.	External interrupt/event GPIO mapping	444
Figure 42.	CRC calculation unit block diagram	455
Figure 43.	FSMC block diagram	461
Figure 44.	FSMC memory banks	464
Figure 45.	Mode1 read access waveforms	471
Figure 46.	Mode1 write access waveforms	472
Figure 47.	ModeA read access waveforms	474
Figure 48.	ModeA write access waveforms	474

Figure 49.	Mode2 and mode B read access waveforms	476
Figure 50.	Mode2 write access waveforms	477
Figure 51.	ModeB write access waveforms	477
Figure 52.	ModeC read access waveforms	479
Figure 53.	ModeC write access waveforms	480
Figure 54.	ModeD read access waveforms	482
Figure 55.	ModeD write access waveforms	483
Figure 56.	Muxed read access waveforms	485
Figure 57.	Muxed write access waveforms	486
Figure 58.	Asynchronous wait during a read access waveforms	488
Figure 59.	Asynchronous wait during a write access waveforms	489
Figure 60.	Wait configuration waveforms	491
Figure 61.	Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)	492
Figure 62.	Synchronous multiplexed write mode waveforms - PSRAM (CRAM)	494
Figure 63.	NAND Flash controller waveforms for common memory access	507
Figure 64.	Access to non 'CE don't care' NAND-Flash	508
Figure 65.	OCTOSPI block diagram for octal communication	519
Figure 66.	OCTOSPI block diagram in quad mode	519
Figure 67.	OCTOSPI block diagram when dual-Flash is enabled	520
Figure 68.	SDR read command in octal mode example	521
Figure 69.	DTR read in octal mode with DQS (Macronix mode) example	524
Figure 70.	SDR write command in Octo-SPI mode example	525
Figure 71.	DTR write in octal mode (macronix mode) example	526
Figure 72.	Example of HyperBus™ read operation	528
Figure 73.	HyperBus™ read operation with initial latency	529
Figure 74.	HyperBus™ write operation with initial latency	529
Figure 75.	HyperBus™ read operation with additional latency	530
Figure 76.	HyperBus™ write operation with additional latency	530
Figure 77.	HyperBus™ write operation with no latency	531
Figure 78.	HyperBus™ read operation page crossing with latency	531
Figure 79.	nCS when CKMODE=0 (T = CLK period)	542
Figure 80.	nCS when CKMODE=1 in SDR mode (T = CLK period)	542
Figure 81.	nCS when CKMODE=1 in DDTR mode (T = CLK period)	543
Figure 82.	nCS when CKMODE=1 with an abort (T = CLK period)	543
Figure 83.	OctoSPI IO Manager block diagram	568
Figure 84.	ADC block diagram	577
Figure 85.	ADC clock scheme	580
Figure 86.	ADC1 connectivity	581
Figure 87.	ADC calibration	584
Figure 88.	Updating the ADC calibration factor	585
Figure 89.	Mixing single-ended and differential channels	585
Figure 90.	Enabling / Disabling the ADC	587
Figure 91.	Bulb mode timing diagram	589
Figure 92.	Analog to digital conversion time	592
Figure 93.	Stopping ongoing regular conversions	593
Figure 94.	Stopping ongoing regular and injected conversions	594
Figure 95.	Injected conversion latency	598
Figure 96.	Example of JSQR queue of context (sequence change)	601
Figure 97.	Example of JSQR queue of context (trigger change)	601
Figure 98.	Example of JSQR queue of context with overflow before conversion	602
Figure 99.	Example of JSQR queue of context with overflow during conversion	602
Figure 100.	Example of JSQR queue of context with empty queue (case JQM=0)	603



Figure 101. Example of JSQR queue of context with empty queue (case JQM=1).	604
Figure 102. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion.	604
Figure 103. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion and a new trigger occurs.	605
Figure 104. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs outside an ongoing conversion	605
Figure 105. Flushing JSQR queue of context by setting JADSTP=1 (JQM=1)	606
Figure 106. Flushing JSQR queue of context by setting ADDIS=1 (JQM=0).	606
Figure 107. Flushing JSQR queue of context by setting ADDIS=1 (JQM=1).	607
Figure 108. Single conversions of a sequence, software trigger	609
Figure 109. Continuous conversion of a sequence, software trigger	609
Figure 110. Single conversions of a sequence, hardware trigger	610
Figure 111. Continuous conversions of a sequence, hardware trigger	610
Figure 112. Right alignment (offset disabled, unsigned value)	612
Figure 113. Right alignment (offset enabled, signed value).	613
Figure 114. Left alignment (offset disabled, unsigned value)	613
Figure 115. Left alignment (offset enabled, signed value).	614
Figure 116. Example of overrun (OVR)	615
Figure 117. AUTODLY=1, regular conversion in continuous mode, software trigger	618
Figure 118. AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=0; JDISCEN=0)	619
Figure 119. AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=1, JDISCEN=1)	620
Figure 120. AUTODLY=1, regular continuous conversions interrupted by injected conversions	621
Figure 121. AUTODLY=1 in auto- injected mode (JAUTO=1).	621
Figure 122. Analog watchdog guarded area	622
Figure 123. ADCy_AWDx_OUT signal generation (on all regular channels).	624
Figure 124. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by software)	625
Figure 125. ADCy_AWDx_OUT signal generation (on a single regular channel)	625
Figure 126. ADCy_AWDx_OUT signal generation (on all injected channels)	625
Figure 127. 20-bit to 16-bit result truncation	626
Figure 128. Numerical example with 5-bit shift and rounding	626
Figure 129. Triggered regular oversampling mode (TROVS bit = 1).	628
Figure 130. Regular oversampling modes (4x ratio)	629
Figure 131. Regular and injected oversampling modes used simultaneously.	630
Figure 132. Triggered regular oversampling with injection	630
Figure 133. Oversampling in auto-injected mode	631
Figure 134. Temperature sensor channel block diagram	632
Figure 135. VBAT channel block diagram	633
Figure 136. VREFINT channel block diagram	633
Figure 137. Dual-channel DAC block diagram	670
Figure 138. Data registers in single DAC channel mode.	672
Figure 139. Data registers in dual DAC channel mode	672
Figure 140. Timing diagram for conversion with trigger disabled TEN = 0	673
Figure 141. DAC LFSR register calculation algorithm	676
Figure 142. DAC conversion (SW trigger enabled) with LFSR wave generation	676
Figure 143. DAC triangle wave generation	677
Figure 144. DAC conversion (SW trigger enabled) with triangle wave generation	677
Figure 145. DAC Sample and Hold mode phase diagram.	679
Figure 146. DCMI block diagram	705

Figure 147. Top-level block diagram	705
Figure 148. DCMI signal waveforms	706
Figure 149. Timing diagram	708
Figure 150. Frame capture waveforms in snapshot mode	710
Figure 151. Frame capture waveforms in continuous grab mode	711
Figure 152. Coordinates and size of the window after cropping	711
Figure 153. Data capture waveforms	712
Figure 154. Pixel raster scan order	713
Figure 155. Comparators block diagram	733
Figure 156. Window mode	735
Figure 157. Comparator hysteresis	736
Figure 158. Comparator output blanking	736
Figure 159. Standalone mode: external gain setting mode	746
Figure 160. Follower configuration	747
Figure 161. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used	748
Figure 162. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering	749
Figure 163. Single DFSDM block diagram	760
Figure 164. Input channel pins redirection	764
Figure 165. Channel transceiver timing diagrams	766
Figure 166. Clock absence timing diagram for SPI	767
Figure 167. Clock absence timing diagram for Manchester coding	768
Figure 168. First conversion for Manchester coding (Manchester synchronization)	770
Figure 169. DFSDM_CHyDATINR registers operation modes and assignment	774
Figure 170. Example: Sinc3 filter response	776
Figure 171. LTDC block diagram	817
Figure 172. LCD-TFT synchronous timings	820
Figure 173. Layer window programmable parameters	823
Figure 174. Blending two layers with background	825
Figure 175. Interrupt events	827
Figure 176. DSI Host block diagram	853
Figure 177. DSI Host architecture	854
Figure 178. Flow to update the LTDC interface configuration using shadow registers	859
Figure 179. Immediate update procedure	860
Figure 180. Configuration update during the transmission of a frame	860
Figure 181. Adapted command mode usage flow	862
Figure 182. 24 bpp APB pixel to byte organization	867
Figure 183. 18 bpp APB pixel to byte organization	867
Figure 184. 16 bpp APB pixel to byte organization	868
Figure 185. 12 bpp APB pixel to byte organization	868
Figure 186. 8 bpp APB pixel to byte organization	868
Figure 187. Timing of PRESP_TO after a bus turn-around	871
Figure 188. Timing of PRESP_TO after a Read Request (HS or LP)	872
Figure 189. Timing of PRESP_TO after a Write Request (HS or LP)	873
Figure 190. Effect of Prep mode at 1	874
Figure 191. Command transmission periods within the image area	875
Figure 192. Transmission of commands on the last line of a frame	876
Figure 193. LPSIZE for Non-Burst with sync pulses	877
Figure 194. LPSIZE for Burst or Non-Burst with sync events	878
Figure 195. VLPSIZE for Non-Burst with sync pulses	879
Figure 196. VLPSIZE for Non-Burst with sync events	879
Figure 197. VLPSIZE for Burst mode	879

Figure 198. Location of LPSIZE and VLPSIZE in the image area . . . . .	881
Figure 199. Clock lane and data lanes in HS . . . . .	883
Figure 200. Clock lane in HS and data lanes in LP . . . . .	883
Figure 201. Clock lane and data lanes in LP . . . . .	883
Figure 202. Command transmission by the generic interface . . . . .	884
Figure 203. Vertical color bar mode . . . . .	886
Figure 204. Horizontal color bar mode . . . . .	886
Figure 205. RGB888 BER testing pattern . . . . .	887
Figure 206. Vertical pattern (103x15) . . . . .	888
Figure 207. Horizontal pattern (103x15) . . . . .	888
Figure 208. PLL block diagram . . . . .	892
Figure 209. Error sources . . . . .	895
Figure 210. Video packet transmission configuration flow diagram . . . . .	906
Figure 211. Programming sequence to send a test pattern . . . . .	908
Figure 212. Frame configuration registers . . . . .	909
Figure 213. TSC block diagram . . . . .	973
Figure 214. Surface charge transfer analog I/O group structure . . . . .	974
Figure 215. Sampling capacitor voltage variation . . . . .	975
Figure 216. Charge transfer acquisition sequence . . . . .	976
Figure 217. Spread spectrum variation principle . . . . .	977
Figure 218. RNG block diagram . . . . .	992
Figure 219. Entropy source model . . . . .	993
Figure 220. RNG initialization overview . . . . .	996
Figure 221. AES block diagram . . . . .	1006
Figure 222. ECB encryption and decryption principle . . . . .	1008
Figure 223. CBC encryption and decryption principle . . . . .	1009
Figure 224. CTR encryption and decryption principle . . . . .	1010
Figure 225. GCM encryption and authentication principle . . . . .	1011
Figure 226. GMAC authentication principle . . . . .	1011
Figure 227. CCM encryption and authentication principle . . . . .	1012
Figure 228. STM32 cryptolib AES flowchart examples . . . . .	1013
Figure 229. STM32 cryptolib AES flowchart examples (continued) . . . . .	1014
Figure 230. Encryption key derivation for ECB/CBC decryption (Mode 2) . . . . .	1017
Figure 231. Example of suspend mode management . . . . .	1018
Figure 232. ECB encryption . . . . .	1019
Figure 233. ECB decryption . . . . .	1019
Figure 234. CBC encryption . . . . .	1020
Figure 235. CBC decryption . . . . .	1020
Figure 236. ECB/CBC encryption (Mode 1) . . . . .	1021
Figure 237. ECB/CBC decryption (Mode 3) . . . . .	1022
Figure 238. Message construction in CTR mode . . . . .	1024
Figure 239. CTR encryption . . . . .	1024
Figure 240. CTR decryption . . . . .	1025
Figure 241. Message construction in GCM . . . . .	1026
Figure 242. GCM authenticated encryption . . . . .	1028
Figure 243. Message construction in GMAC mode . . . . .	1032
Figure 244. GMAC authentication mode . . . . .	1032
Figure 245. Message construction in CCM mode . . . . .	1033
Figure 246. CCM mode authenticated encryption . . . . .	1035
Figure 247. 128-bit block construction with respect to data swap . . . . .	1040
Figure 248. DMA transfer of a 128-bit data block during input phase . . . . .	1042
Figure 249. DMA transfer of a 128-bit data block during output phase . . . . .	1043



Figure 250. AES interrupt signal generation . . . . .	1044
Figure 251. HASH block diagram . . . . .	1060
Figure 252. Message data swapping feature . . . . .	1062
Figure 253. HASH save/restore mechanism . . . . .	1067
Figure 254. HASH interrupt mapping diagram . . . . .	1069
Figure 255. Advanced-control timer block diagram . . . . .	1084
Figure 256. Counter timing diagram with prescaler division change from 1 to 2 . . . . .	1086
Figure 257. Counter timing diagram with prescaler division change from 1 to 4 . . . . .	1086
Figure 258. Counter timing diagram, internal clock divided by 1 . . . . .	1088
Figure 259. Counter timing diagram, internal clock divided by 2 . . . . .	1088
Figure 260. Counter timing diagram, internal clock divided by 4 . . . . .	1089
Figure 261. Counter timing diagram, internal clock divided by N . . . . .	1089
Figure 262. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) . . . . .	1090
Figure 263. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) . . . . .	1090
Figure 264. Counter timing diagram, internal clock divided by 1 . . . . .	1092
Figure 265. Counter timing diagram, internal clock divided by 2 . . . . .	1092
Figure 266. Counter timing diagram, internal clock divided by 4 . . . . .	1093
Figure 267. Counter timing diagram, internal clock divided by N . . . . .	1093
Figure 268. Counter timing diagram, update event when repetition counter is not used . . . . .	1094
Figure 269. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 . . . . .	1095
Figure 270. Counter timing diagram, internal clock divided by 2 . . . . .	1096
Figure 271. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 . . . . .	1096
Figure 272. Counter timing diagram, internal clock divided by N . . . . .	1097
Figure 273. Counter timing diagram, update event with ARPE=1 (counter underflow) . . . . .	1097
Figure 274. Counter timing diagram, Update event with ARPE=1 (counter overflow) . . . . .	1098
Figure 275. Update rate examples depending on mode and TIMx_RCR register settings . . . . .	1099
Figure 276. External trigger input block . . . . .	1100
Figure 277. TIM1 ETR input circuitry . . . . .	1100
Figure 278. TIM8 ETR input circuitry . . . . .	1100
Figure 279. Control circuit in normal mode, internal clock divided by 1 . . . . .	1101
Figure 280. TI2 external clock connection example . . . . .	1102
Figure 281. Control circuit in external clock mode 1 . . . . .	1103
Figure 282. External trigger input block . . . . .	1103
Figure 283. Control circuit in external clock mode 2 . . . . .	1104
Figure 284. Capture/compare channel (example: channel 1 input stage) . . . . .	1105
Figure 285. Capture/compare channel 1 main circuit . . . . .	1106
Figure 286. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3) . . . . .	1106
Figure 287. Output stage of capture/compare channel (channel 4) . . . . .	1107
Figure 288. Output stage of capture/compare channel (channel 5, idem ch. 6) . . . . .	1107
Figure 289. PWM input mode timing . . . . .	1109
Figure 290. Output compare mode, toggle on OC1 . . . . .	1111
Figure 291. Edge-aligned PWM waveforms (ARR=8) . . . . .	1112
Figure 292. Center-aligned PWM waveforms (ARR=8) . . . . .	1113
Figure 293. Generation of 2 phase-shifted PWM signals with 50% duty cycle . . . . .	1115
Figure 294. Combined PWM mode on channel 1 and 3 . . . . .	1116
Figure 295. 3-phase combined PWM signals with multiple trigger pulses per period . . . . .	1117
Figure 296. Complementary output with dead-time insertion . . . . .	1118
Figure 297. Dead-time waveforms with delay greater than the negative pulse . . . . .	1118
Figure 298. Dead-time waveforms with delay greater than the positive pulse . . . . .	1119
Figure 299. Break and Break2 circuitry overview . . . . .	1121
Figure 300. Various output behavior in response to a break event on BRK (OSS1 = 1) . . . . .	1123
Figure 301. PWM output state following BRK and BRK2 pins assertion (OSS1=1) . . . . .	1124

Figure 302. PWM output state following BRK assertion (OSSI=0)	1125
Figure 303. Output redirection (BRK2 request not represented)	1126
Figure 304. Clearing TIMx OCxREF	1127
Figure 305. 6-step generation, COM example (OSSR=1)	1128
Figure 306. Example of one pulse mode	1129
Figure 307. Retriggerable one pulse mode	1131
Figure 308. Example of counter operation in encoder interface mode	1132
Figure 309. Example of encoder interface mode with TI1FP1 polarity inverted	1133
Figure 310. Measuring time interval between edges on 3 signals	1134
Figure 311. Example of Hall sensor interface	1136
Figure 312. Control circuit in reset mode	1137
Figure 313. Control circuit in Gated mode	1138
Figure 314. Control circuit in trigger mode	1139
Figure 315. Control circuit in external clock mode 2 + trigger mode	1140
Figure 316. General-purpose timer block diagram	1190
Figure 317. Counter timing diagram with prescaler division change from 1 to 2	1192
Figure 318. Counter timing diagram with prescaler division change from 1 to 4	1192
Figure 319. Counter timing diagram, internal clock divided by 1	1193
Figure 320. Counter timing diagram, internal clock divided by 2	1194
Figure 321. Counter timing diagram, internal clock divided by 4	1194
Figure 322. Counter timing diagram, internal clock divided by N	1195
Figure 323. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)	1195
Figure 324. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)	1196
Figure 325. Counter timing diagram, internal clock divided by 1	1197
Figure 326. Counter timing diagram, internal clock divided by 2	1197
Figure 327. Counter timing diagram, internal clock divided by 4	1198
Figure 328. Counter timing diagram, internal clock divided by N	1198
Figure 329. Counter timing diagram, Update event when repetition counter is not used	1199
Figure 330. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6	1200
Figure 331. Counter timing diagram, internal clock divided by 2	1201
Figure 332. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36	1201
Figure 333. Counter timing diagram, internal clock divided by N	1202
Figure 334. Counter timing diagram, Update event with ARPE=1 (counter underflow)	1202
Figure 335. Counter timing diagram, Update event with ARPE=1 (counter overflow)	1203
Figure 336. Control circuit in normal mode, internal clock divided by 1	1204
Figure 337. TI2 external clock connection example	1204
Figure 338. Control circuit in external clock mode 1	1205
Figure 339. External trigger input block	1206
Figure 340. Control circuit in external clock mode 2	1207
Figure 341. Capture/Compare channel (example: channel 1 input stage)	1208
Figure 342. Capture/Compare channel 1 main circuit	1208
Figure 343. Output stage of Capture/Compare channel (channel 1)	1209
Figure 344. PWM input mode timing	1211
Figure 345. Output compare mode, toggle on OC1	1213
Figure 346. Edge-aligned PWM waveforms (ARR=8)	1214
Figure 347. Center-aligned PWM waveforms (ARR=8)	1215
Figure 348. Generation of 2 phase-shifted PWM signals with 50% duty cycle	1216
Figure 349. Combined PWM mode on channels 1 and 3	1218
Figure 350. Clearing TIMx OCxREF	1219
Figure 351. Example of one-pulse mode	1220
Figure 352. Retriggerable one pulse mode	1222

Figure 353. Example of counter operation in encoder interface mode	1223
Figure 354. Example of encoder interface mode with TI1FP1 polarity inverted	1224
Figure 355. Control circuit in reset mode	1225
Figure 356. Control circuit in gated mode	1226
Figure 357. Control circuit in trigger mode	1227
Figure 358. Control circuit in external clock mode 2 + trigger mode	1228
Figure 359. Master/Slave timer example	1228
Figure 360. Gating TIM2 with OC1REF of TIM3	1229
Figure 361. Gating TIM2 with Enable of TIM3	1230
Figure 362. Triggering TIM2 with update of TIM3	1231
Figure 363. Triggering TIM2 with Enable of TIM3	1231
Figure 364. Triggering TIM3 and TIM2 with TIM3 TI1 input.	1232
Figure 365. TIM15 block diagram	1262
Figure 366. TIM16/TIM17 block diagram	1263
Figure 367. Counter timing diagram with prescaler division change from 1 to 2	1265
Figure 368. Counter timing diagram with prescaler division change from 1 to 4	1265
Figure 369. Counter timing diagram, internal clock divided by 1	1267
Figure 370. Counter timing diagram, internal clock divided by 2	1267
Figure 371. Counter timing diagram, internal clock divided by 4	1268
Figure 372. Counter timing diagram, internal clock divided by N	1268
Figure 373. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded).	1269
Figure 374. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded).	1269
Figure 375. Update rate examples depending on mode and TIMx_RCR register settings	1271
Figure 376. Control circuit in normal mode, internal clock divided by 1	1272
Figure 377. TI2 external clock connection example	1272
Figure 378. Control circuit in external clock mode 1	1273
Figure 379. Capture/compare channel (example: channel 1 input stage)	1274
Figure 380. Capture/compare channel 1 main circuit	1274
Figure 381. Output stage of capture/compare channel (channel 1)	1275
Figure 382. Output stage of capture/compare channel (channel 2 for TIM15)	1275
Figure 383. PWM input mode timing	1277
Figure 384. Output compare mode, toggle on OC1	1279
Figure 385. Edge-aligned PWM waveforms (ARR=8)	1280
Figure 386. Combined PWM mode on channel 1 and 2	1281
Figure 387. Complementary output with dead-time insertion.	1282
Figure 388. Dead-time waveforms with delay greater than the negative pulse.	1283
Figure 389. Dead-time waveforms with delay greater than the positive pulse.	1283
Figure 390. Break circuitry overview	1285
Figure 391. Output behavior in response to a break	1287
Figure 392. Output redirection	1289
Figure 393. Example of one pulse mode.	1290
Figure 394. Retriggerable one pulse mode	1292
Figure 395. Measuring time interval between edges on 2 signals	1293
Figure 396. Control circuit in reset mode	1294
Figure 397. Control circuit in gated mode	1295
Figure 398. Control circuit in trigger mode	1296
Figure 399. Basic timer block diagram.	1347
Figure 400. Counter timing diagram with prescaler division change from 1 to 2	1349
Figure 401. Counter timing diagram with prescaler division change from 1 to 4	1349
Figure 402. Counter timing diagram, internal clock divided by 1	1350

Figure 403. Counter timing diagram, internal clock divided by 2	1351
Figure 404. Counter timing diagram, internal clock divided by 4	1351
Figure 405. Counter timing diagram, internal clock divided by N	1352
Figure 406. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)	1352
Figure 407. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)	1353
Figure 408. Control circuit in normal mode, internal clock divided by 1	1354
Figure 409. Low-power timer block diagram	1361
Figure 410. Glitch filter timing diagram	1363
Figure 411. LPTIM output waveform, single counting mode configuration	1365
Figure 412. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set)	1365
Figure 413. LPTIM output waveform, Continuous counting mode configuration	1366
Figure 414. Waveform generation	1367
Figure 415. Encoder mode counting sequence	1371
Figure 416. IRTIM internal hardware connections with TIM16 and TIM17	1384
Figure 417. Independent watchdog block diagram	1385
Figure 418. Watchdog block diagram	1395
Figure 419. Window watchdog timing diagram	1396
Figure 420. RTC block diagram	1403
Figure 421. I2C block diagram	1448
Figure 422. I2C bus protocol	1450
Figure 423. Setup and hold timings	1452
Figure 424. I2C initialization flowchart	1455
Figure 425. Data reception	1456
Figure 426. Data transmission	1457
Figure 427. Slave initialization flowchart	1460
Figure 428. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=0	1462
Figure 429. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=1	1463
Figure 430. Transfer bus diagrams for I2C slave transmitter	1464
Figure 431. Transfer sequence flowchart for slave receiver with NOSTRETCH=0	1465
Figure 432. Transfer sequence flowchart for slave receiver with NOSTRETCH=1	1466
Figure 433. Transfer bus diagrams for I2C slave receiver	1466
Figure 434. Master clock generation	1468
Figure 435. Master initialization flowchart	1470
Figure 436. 10-bit address read access with HEAD10R=0	1470
Figure 437. 10-bit address read access with HEAD10R=1	1471
Figure 438. Transfer sequence flowchart for I2C master transmitter for N≤255 bytes	1472
Figure 439. Transfer sequence flowchart for I2C master transmitter for N>255 bytes	1473
Figure 440. Transfer bus diagrams for I2C master transmitter	1474
Figure 441. Transfer sequence flowchart for I2C master receiver for N≤255 bytes	1476
Figure 442. Transfer sequence flowchart for I2C master receiver for N >255 bytes	1477
Figure 443. Transfer bus diagrams for I2C master receiver	1478
Figure 444. Timeout intervals for $t_{LOW:SEXT}$ , $t_{LOW:MEXT}$	1482
Figure 445. Transfer sequence flowchart for SMBus slave transmitter N bytes + PEC	1486
Figure 446. Transfer bus diagrams for SMBus slave transmitter (SBC=1)	1487
Figure 447. Transfer sequence flowchart for SMBus slave receiver N Bytes + PEC	1488
Figure 448. Bus transfer diagrams for SMBus slave receiver (SBC=1)	1489
Figure 449. Bus transfer diagrams for SMBus master transmitter	1490
Figure 450. Bus transfer diagrams for SMBus master receiver	1492
Figure 451. I2C interrupt mapping diagram	1497

Figure 452. USART block diagram . . . . .	1518
Figure 453. Word length programming . . . . .	1521
Figure 454. Configurable stop bits . . . . .	1523
Figure 455. TC/TXE behavior when transmitting . . . . .	1525
Figure 456. Start bit detection when oversampling by 16 or 8 . . . . .	1526
Figure 457. usart_ker_ck clock divider block diagram . . . . .	1529
Figure 458. Data sampling when oversampling by 16 . . . . .	1531
Figure 459. Data sampling when oversampling by 8 . . . . .	1531
Figure 460. Mute mode using Idle line detection . . . . .	1538
Figure 461. Mute mode using address mark detection . . . . .	1539
Figure 462. Break detection in LIN mode (11-bit break length - LBDL bit is set) . . . . .	1542
Figure 463. Break detection in LIN mode vs. Framing error detection . . . . .	1543
Figure 464. USART example of synchronous master transmission . . . . .	1544
Figure 465. USART data clock timing diagram in synchronous master mode (M bits =00) . . . . .	1544
Figure 466. USART data clock timing diagram in synchronous master mode (M bits = '01') . . . . .	1545
Figure 467. USART data clock timing diagram in synchronous slave mode (M bits =00) . . . . .	1546
Figure 468. ISO 7816-3 asynchronous protocol . . . . .	1548
Figure 469. Parity error detection using the 1.5 stop bits . . . . .	1550
Figure 470. IrDA SIR ENDEC block diagram . . . . .	1554
Figure 471. IrDA data modulation (3/16) - Normal mode . . . . .	1554
Figure 472. Transmission using DMA . . . . .	1556
Figure 473. Reception using DMA . . . . .	1557
Figure 474. Hardware flow control between 2 USARTs . . . . .	1557
Figure 475. RS232 RTS flow control . . . . .	1558
Figure 476. RS232 CTS flow control . . . . .	1559
Figure 477. Wakeup event verified (wakeup event = address match, FIFO disabled) . . . . .	1562
Figure 478. Wakeup event not verified (wakeup event = address match, FIFO disabled) . . . . .	1562
Figure 479. LPUART block diagram . . . . .	1596
Figure 480. LPUART word length programming . . . . .	1598
Figure 481. Configurable stop bits . . . . .	1600
Figure 482. TC/TXE behavior when transmitting . . . . .	1602
Figure 483. lpuart_ker_ck clock divider block diagram . . . . .	1605
Figure 484. Mute mode using Idle line detection . . . . .	1609
Figure 485. Mute mode using address mark detection . . . . .	1610
Figure 486. Transmission using DMA . . . . .	1613
Figure 487. Reception using DMA . . . . .	1614
Figure 488. Hardware flow control between 2 LPUARTs . . . . .	1614
Figure 489. RS232 RTS flow control . . . . .	1615
Figure 490. RS232 CTS flow control . . . . .	1616
Figure 491. Wakeup event verified (wakeup event = address match, FIFO disabled) . . . . .	1619
Figure 492. Wakeup event not verified (wakeup event = address match, FIFO disabled) . . . . .	1619
Figure 493. SPI block diagram . . . . .	1641
Figure 494. Full-duplex single master/ single slave application . . . . .	1642
Figure 495. Half-duplex single master/ single slave application . . . . .	1643
Figure 496. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode) . . . . .	1644



Figure 497. Master and three independent slaves . . . . .	1645
Figure 498. Multi-master application . . . . .	1646
Figure 499. Hardware/software slave select management . . . . .	1647
Figure 500. Data clock timing diagram . . . . .	1648
Figure 501. Data alignment when data length is not equal to 8-bit or 16-bit . . . . .	1649
Figure 502. Packing data in FIFO for transmission and reception . . . . .	1653
Figure 503. Master full-duplex communication . . . . .	1656
Figure 504. Slave full-duplex communication . . . . .	1657
Figure 505. Master full-duplex communication with CRC . . . . .	1658
Figure 506. Master full-duplex communication in packed mode . . . . .	1659
Figure 507. NSSP pulse generation in Motorola SPI master mode . . . . .	1662
Figure 508. TI mode transfer . . . . .	1663
Figure 509. SAI functional block diagram . . . . .	1677
Figure 510. Audio frame . . . . .	1681
Figure 511. FS role is start of frame + channel side identification (FSDEF = TRIS = 1) . . . . .	1683
Figure 512. FS role is start of frame (FSDEF = 0) . . . . .	1684
Figure 513. Slot size configuration with FBOFF = 0 in SAI_xSLOTR . . . . .	1685
Figure 514. First bit offset . . . . .	1685
Figure 515. Audio block clock generator overview . . . . .	1686
Figure 516. PDM typical connection and timing . . . . .	1690
Figure 517. Detailed PDM interface block diagram . . . . .	1691
Figure 518. Start-up sequence . . . . .	1692
Figure 519. SAI_ADR format in TDM, 32-bit slot width . . . . .	1693
Figure 520. SAI_ADR format in TDM, 16-bit slot width . . . . .	1694
Figure 521. SAI_ADR format in TDM, 8-bit slot width . . . . .	1695
Figure 522. AC'97 audio frame . . . . .	1698
Figure 523. Example of typical AC'97 configuration on devices featuring at least 2 embedded SAIs (three external AC'97 decoders) . . . . .	1699
Figure 524. SPDIF format . . . . .	1700
Figure 525. SAI_xDR register ordering . . . . .	1701
Figure 526. Data companding hardware in an audio block in the SAI . . . . .	1704
Figure 527. Tristate strategy on SD output line on an inactive slot . . . . .	1706
Figure 528. Tristate on output data line in a protocol like I2S . . . . .	1707
Figure 529. Overrun detection error . . . . .	1708
Figure 530. FIFO underrun event . . . . .	1708
Figure 531. SDMMC "no response" and "no data" operations . . . . .	1743
Figure 532. SDMMC (multiple) block read operation . . . . .	1743
Figure 533. SDMMC (multiple) block write operation . . . . .	1743
Figure 534. SDMMC (sequential) stream read operation . . . . .	1744
Figure 535. SDMMC (sequential) stream write operation . . . . .	1744
Figure 536. SDMMC block diagram . . . . .	1745
Figure 537. SDMMC Command and data phase relation . . . . .	1747
Figure 538. Control unit . . . . .	1749
Figure 539. Command/Response path . . . . .	1750
Figure 540. Command path state machine (CPSM) . . . . .	1751
Figure 541. Data path . . . . .	1757
Figure 542. DDR mode data packet clocking . . . . .	1758
Figure 543. DDR mode CRC status / boot acknowledgment clocking . . . . .	1758
Figure 544. Data path state machine (DPSM) . . . . .	1759
Figure 545. CLKMUX unit . . . . .	1769
Figure 546. Asynchronous interrupt generation . . . . .	1773
Figure 547. Synchronous interrupt period data read . . . . .	1774

Figure 548. Synchronous interrupt period data write . . . . .	1774
Figure 549. Asynchronous interrupt period data read . . . . .	1775
Figure 550. Asynchronous interrupt period data write . . . . .	1776
Figure 551. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25. . . . .	1779
Figure 552. Clock stop with SDMMC_CK for DDR50, SDR50. . . . .	1779
Figure 553. ReadWait with SDMMC_CK < 50 MHz. . . . .	1780
Figure 554. ReadWait with SDMMC_CK > 50 MHz. . . . .	1780
Figure 555. CMD12 stream timing . . . . .	1783
Figure 556. CMD5 Sleep Awake procedure . . . . .	1785
Figure 557. Normal boot mode operation . . . . .	1787
Figure 558. Alternative boot mode operation . . . . .	1788
Figure 559. Command response R1b busy signaling . . . . .	1789
Figure 560. SDMMC state control . . . . .	1790
Figure 561. Card cycle power / power up diagram . . . . .	1791
Figure 562. CMD11 signal voltage switch sequence . . . . .	1792
Figure 563. Voltage switch transceiver typical application. . . . .	1794
Figure 564. CAN network topology . . . . .	1820
Figure 565. Dual-CAN block diagram . . . . .	1821
Figure 566. bxCAN operating modes. . . . .	1823
Figure 567. bxCAN in silent mode . . . . .	1824
Figure 568. bxCAN in loop back mode . . . . .	1824
Figure 569. bxCAN in combined mode . . . . .	1825
Figure 570. Transmit mailbox states . . . . .	1827
Figure 571. Receive FIFO states . . . . .	1828
Figure 572. Filter bank scale configuration - register organization . . . . .	1831
Figure 573. Example of filter numbering . . . . .	1832
Figure 574. Filtering mechanism - example . . . . .	1833
Figure 575. CAN error state diagram. . . . .	1834
Figure 576. Bit timing . . . . .	1836
Figure 577. CAN frames . . . . .	1837
Figure 578. Event flags and interrupt generation. . . . .	1838
Figure 579. CAN mailbox registers . . . . .	1850
Figure 580. OTG full-speed block diagram . . . . .	1868
Figure 581. OTG_FS A-B device connection. . . . .	1870
Figure 582. USB_FS peripheral-only connection. . . . .	1872
Figure 583. USB_FS host-only connection . . . . .	1876
Figure 584. SOF connectivity (SOF trigger output to TIM and ITR1 connection) . . . . .	1880
Figure 585. Updating OTG_HFIR dynamically (RLDCTRL = 0) . . . . .	1882
Figure 586. Device-mode FIFO address mapping and AHB FIFO access mapping . . . . .	1883
Figure 587. Host-mode FIFO address mapping and AHB FIFO access mapping. . . . .	1884
Figure 588. Interrupt hierarchy. . . . .	1888
Figure 589. Transmit FIFO write task . . . . .	1972
Figure 590. Receive FIFO read task . . . . .	1973
Figure 591. Normal bulk/control OUT/SETUP . . . . .	1974
Figure 592. Bulk/control IN transactions . . . . .	1978
Figure 593. Normal interrupt OUT . . . . .	1981
Figure 594. Normal interrupt IN . . . . .	1986
Figure 595. Isochronous OUT transactions . . . . .	1988
Figure 596. Isochronous IN transactions . . . . .	1991
Figure 597. Receive FIFO packet read . . . . .	1995
Figure 598. Processing a SETUP packet . . . . .	1997
Figure 599. Bulk OUT transaction . . . . .	2004

---

Figure 600. TRDT max timing case .....	2012
Figure 601. A-device SRP .....	2013
Figure 602. B-device SRP .....	2014
Figure 603. A-device HNP .....	2015
Figure 604. B-device HNP .....	2017
Figure 605. Block diagram of STM32 MCU and Cortex <sup>®</sup> -M4-level debug support .....	2019
Figure 606. SWJ debug port .....	2021
Figure 607. JTAG TAP connections .....	2025
Figure 608. TPIU block diagram .....	2044



# 1 Documentation conventions

## 1.1 General information

The STM32L4Rxxx and STM32L4Sxxx devices have an Arm<sup>®(a)</sup> 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.

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

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

## 1.4 Availability of peripherals

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

## 2 System and memory overview

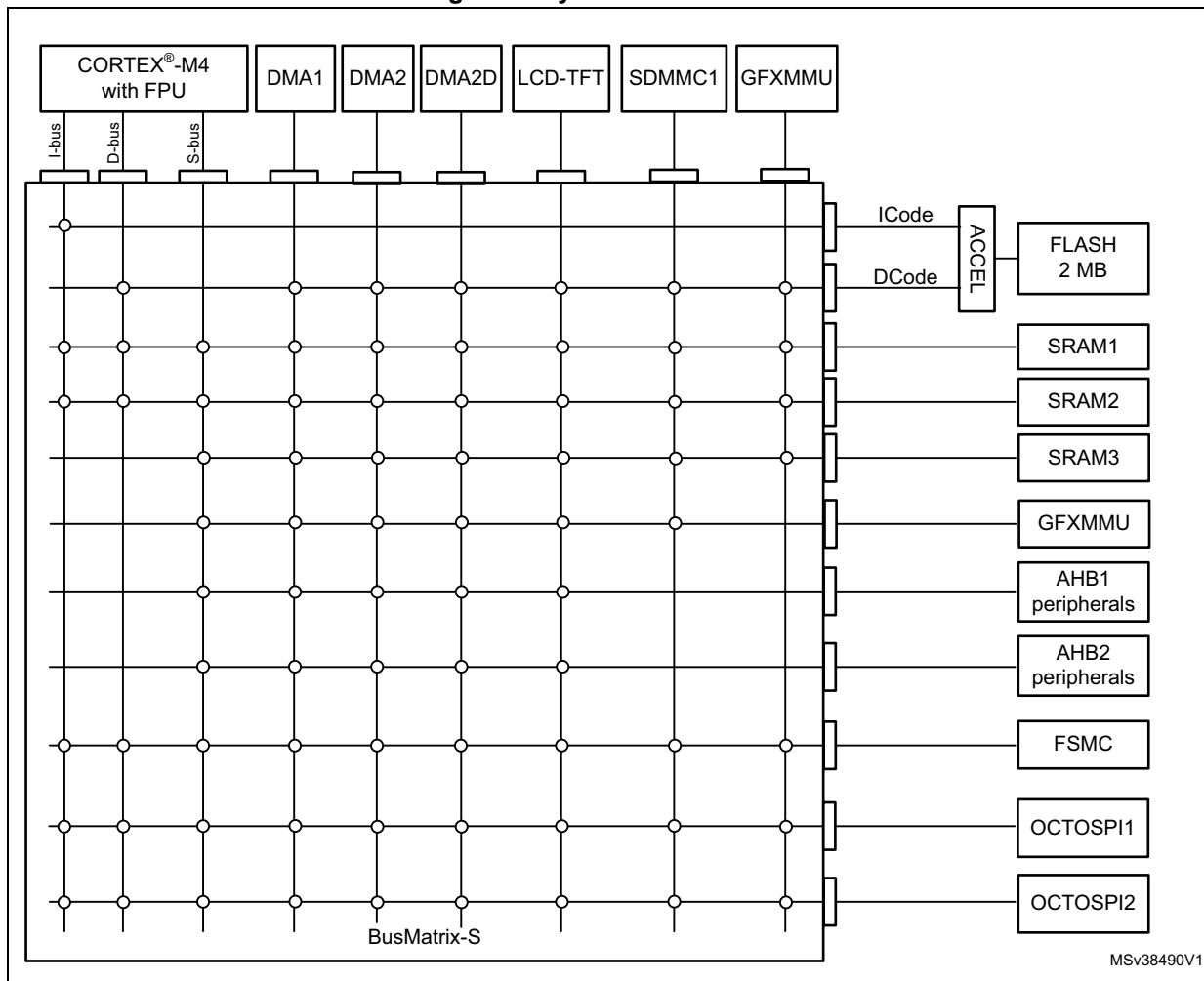
### 2.1 System architecture

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

- Up to nine masters:
  - Cortex<sup>®</sup>-M4 with FPU core I-bus
  - Cortex<sup>®</sup>-M4 with FPU core D-bus
  - Cortex<sup>®</sup>-M4 with FPU core S-bus
  - DMA1
  - DMA2
  - DMA2D (Chrom-Art Accelerator™) memory bus
  - LCD-TFT Controller DMA-bus
  - SDMMC1 bus
  - GFXMMU (Chrom-GRC™) bus
- Up to eleven slaves:
  - Internal Flash memory on the ICode bus
  - Internal Flash memory on DCode bus
  - Internal SRAM1 (192 KB)
  - Internal SRAM2 (64 KB)
  - Internal SRAM3 (384 KB)
  - GFXMMU (Chrom-GRC™)
  - AHB1 peripherals including AHB to APB bridges and APB peripherals (connected to APB1 and APB2)
  - AHB2 peripherals
  - Flexible Memory Controller (FMC)
  - OCTOSPI1
  - OCTOSPI2

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 I-bus

This bus connects the instruction bus of the Cortex<sup>®</sup>-M4 core to the BusMatrix. This bus is used by the core to fetch instructions. The target of this bus is a memory containing code (either internal Flash memory, internal SRAM or external memories through the FMC or OCTOSPIs).

### 2.1.2 D-bus

This bus connects the data bus of the Cortex<sup>®</sup>-M4 core to the BusMatrix. This bus is used by the core for literal load and debug access. The target of this bus is a memory containing code (either internal Flash memory, internal SRAM or external memories through the FMC or OCTOSPIs).

### 2.1.3 S-bus

This bus connects the system bus of the Cortex<sup>®</sup>-M4 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 the internal SRAM, the AHB1 peripherals including the APB1 and APB2

peripherals, the AHB2 peripherals and the external memories through the OCTOSPI or the FMC.

The SRAM2 is also accessible on this bus to allow continuous mapping with SRAM1 and SRAM3.

#### 2.1.4 DMA-bus

This bus connects the AHB master interface of the DMA to the BusMatrix. The targets of this bus are the SRAM1, SRAM2 and SRAM3, the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals and the external memories through the OCTOSPI or the FMC.

#### 2.1.5 DMA2D-bus

This bus connects the AHB master interface of the DMA2D to the BusMatrix. The targets of this bus are the SRAM1, SRAM2 and SRAM3 and external memories through the OCTOSPI or the FMC.

#### 2.1.6 LCD-TFT Controller DMA bus

This bus connects the LCD controller DMA master interface to the BusMatrix. The targets of this bus are data memories: internal SRAMs (SRAM1, SRAM2, SRAM3), internal Flash memory or external memories through FMC or OCTOSPI.

#### 2.1.7 SDMMC Controller DMA bus

This bus connects the SDMMC1 DMA master interface to the BusMatrix. This bus is used only by the SDMMC1 DMA to load/store data from/to memory. The targets of this bus are data memories: internal SRAMs (SRAM1, SRAM2, SRAM3), internal Flash memory or external memories through FMC or OCTOSPI.

#### 2.1.8 GFXMMU-bus

This bus connects the GFXMMU (Chrom-GRC™) AHB master interface to the BusMatrix. The targets of this bus are data memories: internal SRAMs (SRAM1, SRAM2, SRAM3), internal Flash memory or external memories through FMC or OCTOSPI.

#### 2.1.9 BusMatrix

The BusMatrix manages the access arbitration between masters. The arbitration uses a Round Robin algorithm. The BusMatrix is composed of up to nine masters (CPU AHB, system bus, DCode bus, ICode bus, DMA1, DMA2, DMA2D, SDMMC1, LCD-TFT and the GFXMMU) and up to eleven slaves (FLASH, SRAM1, SRAM2, SRAM3, AHB1 (including APB1 and APB2), AHB2, OCTOSTPI1, OCTOSPI2 and FMC).

##### AHB/APB bridges

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

Refer to [Section 2.2.2: Memory map and register boundary addresses on page 82](#) 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**

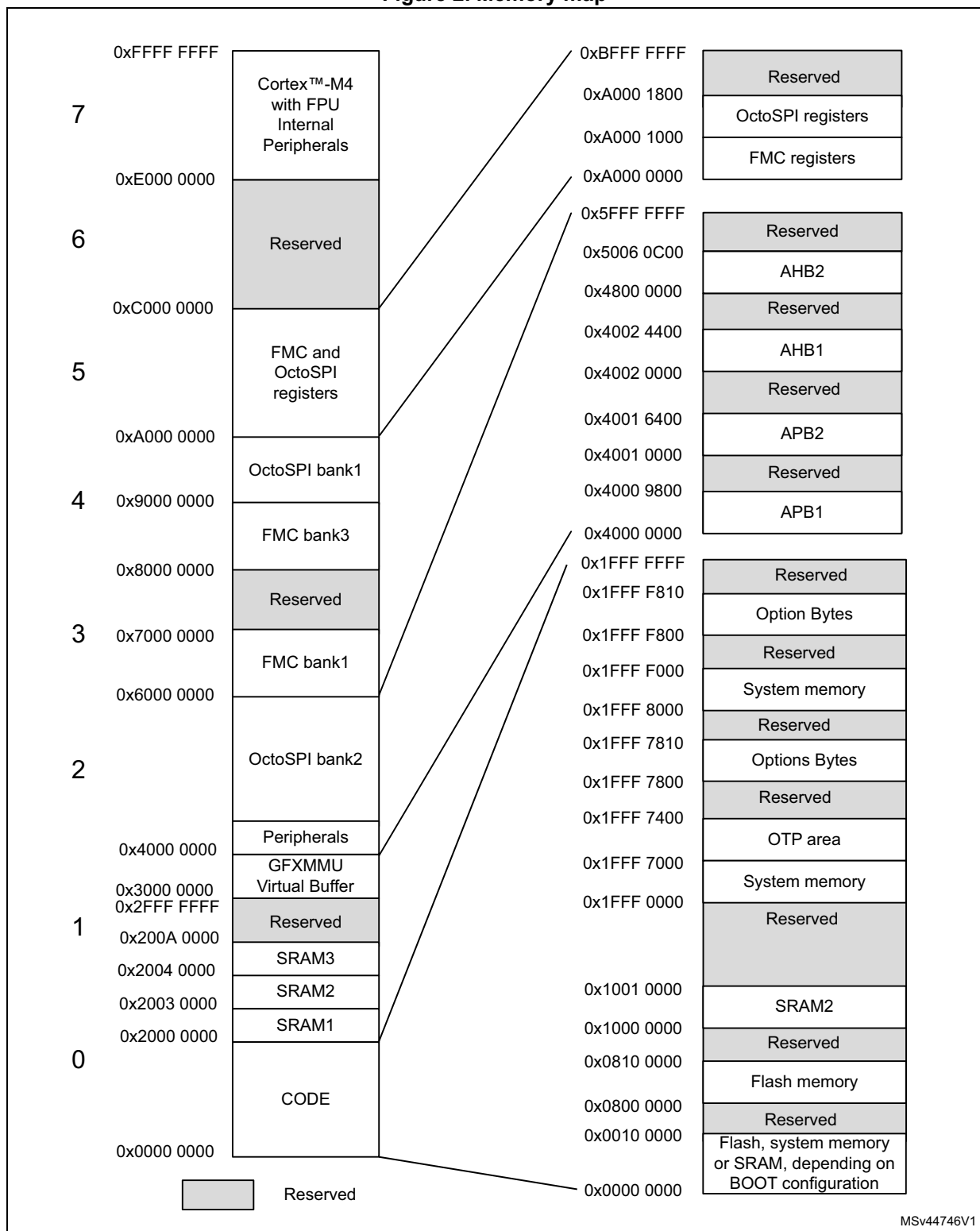
Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space.

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



MSv44746V1



All the memory map areas that are 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.

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

**Table 1. STM32L4Rxxx and STM32L4Sxxx memory map and peripheral register boundary addresses**

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
-	0xA000 1800 - 0xDFFF FFFF	1 KB	Reserved	-
	0xA000 1400 - 0xA000 17FF	1 KB	OCTOSPI2 registers	<a href="#">Section 19.6.23: OCTOSPI register map</a>
	0xA000 1000 - 0xA000 13FF	1 KB	OCTOSPI1 registers	<a href="#">Section 19.6.23: OCTOSPI register map</a>
	0xA000 0400 - 0xA000 0FFF	1 KB	Reserved	-
	0xA000 0000 - 0xA000 03FF	1 KB	FSMC registers	<a href="#">Section 18.7: FSMC register map</a>

**Table 1. STM32L4Rxxx and STM32L4Sxxx memory map and peripheral register boundary addresses (continued)**

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB2	0x5006 2000 - 0x5FFF FFFF	~260 MB	Reserved	-
	0x5006 2400 - 0x5006 27FF	1 KB	SDMMC1	<a href="#">Section 48.8.20: SDMMC register map</a>
	0x5006 2000 - 0x5006 23FF	1 KB	Reserved	-
	0x5006 1C00 - 0x5006 1FFF	1 KB	OCTOSPIM	<a href="#">Section 20.4.2: OctoSPI IO Manager register map</a>
	0x5006 0C00 - 0x5006 1BFF	4 KB	Reserved	-
	0x5006 0800 - 0x5006 0BFF	1 KB	RNG	<a href="#">Section 31.8.4: RNG register map</a>
	0x5006 0400 - 0x5006 07FF	1 KB	HASH	<a href="#">Section 33.6.8: HASH register map</a>
	0x5006 0000 - 0x5006 03FF	1 KB	AES	<a href="#">Section 17.7.18: AES register map</a>
	0x5005 0800 - 0x5005 FFFF	61 KB	Reserved	-
	0x5005 0400 - 0x5005 07FF	1 KB	Reserved	-
	0x5005 0000 - 0x5005 03FF	1 KB	DCMI	<a href="#">Section 23.7.12: DCMI register map</a>
	0x5004 0400 - 0x5004 FFFF	62 KB	Reserved	-
	0x5004 0000 - 0x5004 03FF	1 KB	ADC	<a href="#">Section 21.7.3: ADC register map on page 664</a>
	0x5000 0000 - 0x5003 FFFF	16 KB	OTG_FS	<a href="#">Section 50.15.54: OTG_FS register map</a>
	0x4800 2400 - 0x4FFF FFFF	~127 MB	Reserved	-
	0x4800 2000 - 0x4800 23FF	1 KB	GPIOI	<a href="#">Section 8.4.12: GPIO register map</a>
	0x4800 1C00 - 0x4800 1FFF	1 KB	GPIOH	<a href="#">Section 8.4.12: GPIO register map</a>
	0x4800 1800 - 0x4800 1BFF	1 KB	GPIOG	<a href="#">Section 8.4.12: GPIO register map</a>
	0x4800 1400 - 0x4800 17FF	1 KB	GPIOF	<a href="#">Section 8.4.12: GPIO register map</a>
	0x4800 1000 - 0x4800 13FF	1 KB	GPIOE	<a href="#">Section 8.4.12: GPIO register map</a>
	0x4800 0C00 - 0x4800 0FFF	1 KB	GPIOD	<a href="#">Section 8.4.12: GPIO register map</a>
	0x4800 0800 - 0x4800 0BFF	1 KB	GPIOC	<a href="#">Section 8.4.12: GPIO register map</a>
	0x4800 0400 - 0x4800 07FF	1 KB	GPIOB	<a href="#">Section 8.4.12: GPIO register map</a>
0x4800 0000 - 0x4800 03FF	1 KB	GPIOA	<a href="#">Section 8.4.12: GPIO register map</a>	

**Table 1. STM32L4Rxxx and STM32L4Sxxx memory map and peripheral register boundary addresses (continued)**

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB1	0x4002 F000 - 0x47FF FFFF	~127 MB	Reserved	-
	0x4002 C000 - 0x4002 EFFF	1 KB	GFXMMU	<a href="#">Section 14.5.11: Graphic MMU register map</a>
	0x4002 BC00 - 0x4002 BBFF	1 KB	Reserved	-
	0x4002 B000 - 0x4002 BBFF	3 KB	DMA2D	<a href="#">Section 13.5.21: DMA2D register map</a>
	0x4002 4400 - 0x4002 AFFF	26 KB	Reserved	-
	0x4002 4000 - 0x4002 43FF	1 KB	TSC	<a href="#">Section 30.6.11: TSC register map</a>
	0x4002 3400 - 0x4002 3FFF	1 KB	Reserved	-
	0x4002 3000 - 0x4002 33FF	1 KB	CRC	<a href="#">Section 17.4.6: CRC register map</a>
	0x4002 2400 - 0x4002 2FFF	3 KB	Reserved	-
	0x4002 2000 - 0x4002 23FF	1 KB	FLASH registers	<a href="#">Section 3.7.13: FLASH register map</a>
	0x4002 1400 - 0x4002 1FFF	3 KB	Reserved	-
	0x4002 1000 - 0x4002 13FF	1 KB	RCC	<a href="#">Section 6.4.33: RCC register map</a>
	0x4002 0800 - 0x4002 0FFF	2 KB	Reserved	-
	0x4002 0400 - 0x4002 07FF	1 KB	DMA2	<a href="#">Section 11.6.7: DMA register map and reset values</a>
	0x4002 0800 - 0x4002 0BFF	1 KB	DMAMUX1	<a href="#">Section 12.6.7: DMAMUX register map</a>
	0x4002 0C00 - 0x4002 0FFF	1 KB	Reserved	-
0x4002 0000 - 0x4002 03FF	1 KB	DMA1	<a href="#">Section 11.6.7: DMA register map and reset values</a>	

**Table 1. STM32L4Rxxx and STM32L4Sxxx memory map and peripheral register boundary addresses (continued)**

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB2	0x4001 7400 - 0x4001 FFFF	33 KB	Reserved	-
	0x4001 6C00 - 0x4001 73FF	1 KB	DSIHOST	<a href="#">Section 29.17: DSI Host register map</a>
	0x4001 6800 - 0x4001 6BFF	1 KB	LCD-TFT	<a href="#">Section 28.7.26: LTDC register map</a>
	0x4001 6000 - 0x4001 67FF	2 KB	DFSDM1	<a href="#">Section 27.8.16: DFSDM register map</a>
	0x4001 5C00 - 0x4001 5FFF	1 KB	Reserved	-
	0x4001 5800 - 0x4001 5BFF	1 KB	SAI2	<a href="#">Section 47.5.20: SAI register map</a>
	0x4001 5400 - 0x4001 57FF	1 KB	SAI1	<a href="#">Section 47.5.20: SAI register map</a>
	0x4001 4C00 - 0x4001 53FF	2 KB	Reserved	-
	0x4001 4800 - 0x4001 4BFF	1 KB	TIM17	<a href="#">Section 36.6.20: TIM16/TIM17 register map</a>
	0x4001 4400 - 0x4001 47FF	1 KB	TIM16	<a href="#">Section 36.6.20: TIM16/TIM17 register map</a>
	0x4001 4000 - 0x4001 43FF	1 KB	TIM15	<a href="#">Section 36.5.20: TIM15 register map</a>
	0x4001 3C00 - 0x4001 3FFF	1 KB	Reserved	-
	0x4001 3800 - 0x4001 3BFF	1 KB	USART1	<a href="#">Section 31.8.12: USART register map</a>
	0x4001 3400 - 0x4001 37FF	1 KB	TIM8	<a href="#">Section 34.4.31: TIM8 register map</a>
	0x4001 3000 - 0x4001 33FF	1 KB	SPI1	<a href="#">Section 46.6.8: SPI register map</a>
	0x4001 2C00 - 0x4001 2FFF	1 KB	TIM1	<a href="#">Section 34.4.30: TIM1 register map</a>
	0x4001 2000 - 0x4001 2BFF	3 KB	Reserved	-
	0x4001 1C00 - 0x4001 1FFF	1 KB	FIREWALL	<a href="#">Section 4.4.8: Firewall register map</a>
	0x4001 0800 - 0x4001 1BFF	5 KB	Reserved	-
	0x4001 0400 - 0x4001 07FF	1 KB	EXTI	<a href="#">Section 13.5.13: EXTI register map</a>
0x4001 0200 - 0x4001 03FF	1 KB	COMP	<a href="#">Section 25.6.3: COMP register map</a>	
0x4001 0030 - 0x4001 01FF	1 KB	VREFBUF	<a href="#">Section 24.3.3: VREFBUF register map</a>	
0x4001 0000 - 0x4001 002F	1 KB	SYSCFG	<a href="#">Section 9.2.11: SYSCFG register map</a>	

**Table 1. STM32L4Rxxx and STM32L4Sxxx memory map and peripheral register boundary addresses (continued)**

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB1	0x4000 9800 - 0x4000 FFFF	26 KB	Reserved	-
	0x4000 9400 - 0x4000 97FF	1 KB	LPTIM2	<a href="#">Section 38.7.11: LPTIM register map</a>
	0x4000 8C00 - 0x4000 93FF	3 KB	Reserved	-
	0x4000 8400 - 0x4000 87FF	1 KB	I2C4	<a href="#">Section 43.7.12: I2C register map</a>
	0x4000 8000 - 0x4000 83FF	1 KB	LPUART1	<a href="#">Section 45.5.11: LPUART register map</a>
	0x4000 7C00 - 0x4000 7FFF	1 KB	LPTIM1	<a href="#">Section 38.7.11: LPTIM register map</a>
	0x4000 7800 - 0x4000 7BFF	1 KB	OPAMP	<a href="#">Section 20.5.4: OPAMP register map</a>
	0x4000 7400 - 0x4000 77FF	1 KB	DAC1	<a href="#">Section 22.7.21: DAC register map</a>
	0x4000 7000 - 0x4000 73FF	1 KB	PWR	<a href="#">Section 5.4.20: PWR register map and reset value table</a>
	0x4000 6800 - 0x4000 6FFF	2 KB	Reserved	-
	0x4000 6400 - 0x4000 67FF	1 KB	CAN1	<a href="#">Section 49.9.5: bxCAN register map</a>
	0x4000 6000 - 0x4000 63FF	1 KB	CRS	<a href="#">Section 7.6.5: CRS register map</a>
	0x4000 5C00 - 0x4000 5FFF	1 KB	I2C3	<a href="#">Section 43.7.12: I2C register map</a>
	0x4000 5800 - 0x4000 5BFF	1 KB	I2C2	<a href="#">Section 43.7.12: I2C register map</a>
	0x4000 5400 - 0x4000 57FF	1 KB	I2C1	<a href="#">Section 43.7.12: I2C register map</a>
	0x4000 5000 - 0x4000 53FF	1 KB	UART5	<a href="#">Section 31.8.12: USART register map</a>
	0x4000 4C00 - 0x4000 4FFF	1 KB	UART4	<a href="#">Section 31.8.12: USART register map</a>

**Table 1. STM32L4Rxxx and STM32L4Sxxx memory map and peripheral register boundary addresses (continued)**

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB1	0x4000 4800 - 0x4000 4BFF	1 KB	USART3	<a href="#">Section 31.8.12: USART register map</a>
	0x4000 4400 - 0x4000 47FF	1 KB	USART2	<a href="#">Section 31.8.12: USART register map</a>
	0x4000 4000 - 0x4000 43FF	1 KB	Reserved	-
	0x4000 3C00 - 0x4000 3FFF	1 KB	SPI3	<a href="#">Section 46.6.8: SPI register map</a>
	0x4000 3800 - 0x4000 3BFF	1 KB	SPI2	<a href="#">Section 46.6.8: SPI register map</a>
	0x4000 3400 - 0x4000 37FF	1 KB	Reserved	-
	0x4000 3000 - 0x4000 33FF	1 KB	IWDG	<a href="#">Section 40.4.6: IWDG register map</a>
	0x4000 2C00 - 0x4000 2FFF	1 KB	WWDG	<a href="#">Section 41.4.4: WWDG register map</a>
	0x4000 2800 - 0x4000 2BFF	1 KB	RTC	<a href="#">Section 42.6.21: RTC register map</a>
	0x4000 1800 - 0x4000 23FF	4 KB	Reserved	-
	0x4000 1400 - 0x4000 17FF	1 KB	TIM7	<a href="#">Section 37.4.9: TIM6/TIM7 register map</a>
	0x4000 1000 - 0x4000 13FF	1 KB	TIM6	<a href="#">Section 37.4.9: TIM6/TIM7 register map</a>
	0x4000 0C00 - 0x4000 0FFF	1 KB	TIM5	<a href="#">Section 35.4.23: TIMx register map</a>
	0x4000 0800 - 0x4000 0BFF	1 KB	TIM4	<a href="#">Section 35.4.23: TIMx register map</a>
	0x4000 0400 - 0x4000 07FF	1 KB	TIM3	<a href="#">Section 35.4.23: TIMx register map</a>
0x4000 0000 - 0x4000 03FF	1 KB	TIM2	<a href="#">Section 35.4.23: TIMx register map</a>	

## 2.3 Bit banding

The Cortex<sup>®</sup>-M4 memory map includes two bit-band regions. These regions map each word in an alias region of memory to a bit in a bit-band region of memory. Writing to a word in the alias region has the same effect as a read-modify-write operation on the targeted bit in the bit-band region.

In the STM324Rxxx and STM32L4Sxxx devices both the peripheral registers and the SRAM are mapped to a bit-band region, so that single bit-band write and read operations are allowed. The operations are only available for Cortex<sup>®</sup>-M4 accesses, and not from other bus masters (e.g. DMA).

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

$$\text{bit\_word\_addr} = \text{bit\_band\_base} + (\text{byte\_offset} \times 32) + (\text{bit\_number} \times 4)$$

where:

- *bit\_word\_addr* is the address of the word in the alias memory region that maps to the targeted bit
- *bit\_band\_base* is the starting address of the alias region
- *byte\_offset* is the number of the byte in the bit-band region that contains the targeted bit
- *bit\_number* is the bit position (0-7) of the targeted bit

### Example

The following example shows how to map bit 2 of the byte located at SRAM1 address 0x20000300 to the alias region:

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

Writing to address 0x22006008 has the same effect as a read-modify-write operation on bit 2 of the byte at SRAM1 address 0x20000300.

Reading address 0x22006008 returns the value (0x01 or 0x00) of bit 2 of the byte at SRAM1 address 0x20000300 (0x01: bit set; 0x00: bit reset).

For more information on bit-banding, refer to the *Cortex<sup>®</sup>-M4 programming manual* (see [Related documents on page 1](#)).

## 2.4 Embedded SRAM

The STM32L4Rxxx and STM32L4Sxxx devices feature up to 640 Kbytes SRAM:

- 192 Kbytes SRAM1
- 64 Kbytes SRAM2
- 384 Kbytes SRAM3

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

The CPU can access the SRAM1 through the system bus or through the ICode/DCode buses when boot from SRAM1 is selected or when physical remap is selected ([Section 9.2.1: SYSCFG memory remap register \(SYSCFG\\_MEMRMP\)](#) in the SYSCFG controller). To get the maximum performance on SRAM1 execution, physical remap should be selected (boot or software selection).

Execution can be performed from CCM SRAM with maximum performance without any remap thanks to access through ICode bus.

The SRAM2 is aliased at address 0x2004 0000, offering a continuous address space with the SRAM1 and SRAM3.

### 2.4.1 SRAM2 parity check

The user can enable the SRAM2 parity check using the option bit SRAM2\_PE in the user option byte (refer to [Section 3.4.1: Option bytes description](#)).

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

The parity bits are computed and stored when writing into the SRAM2. Then, they are automatically checked when reading. If one bit fails, an NMI is generated. The same error can also be linked to the BRK\_IN Break input of TIM1/TIM8/TIM15/TIM16/TIM17, with the SPL control bit in the [SYSCFG configuration register 2 \(SYSCFG\\_CFGR2\)](#). The SRAM2 Parity Error flag (SPF) is available in the [SYSCFG configuration register 2 \(SYSCFG\\_CFGR2\)](#).

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

## 2.4.2 SRAM2 Write protection

The SRAM2 can be write protected with a page granularity of 1 Kbyte.

**Table 2. SRAM2 organization**

Page number	Start address	End address
Page 0	0x1000 0000	0x1000 03FF
Page 1	0x1000 0400	0x1000 07FF
Page 2	0x1000 0800	0x1000 0BFF
Page 3	0x1000 0C00	0x1000 0FFF
Page 4	0x1000 1000	0x1000 13FF
Page 5	0x1000 1400	0x1000 17FF
Page 6	0x1000 1800	0x1000 1BFF
Page 7	0x1000 1C00	0x1000 1FFF
Page 8	0x1000 2000	0x1000 23FF
Page 9	0x1000 2400	0x1000 27FF
Page 10	0x1000 2800	0x1000 2BFF
Page 11	0x1000 2C00	0x1000 2FFF
Page 12	0x1000 3000	0x1000 33FF
Page 13	0x1000 3400	0x1000 37FF
Page 14	0x1000 3800	0x1000 3BFF
Page 15	0x1000 3C00	0x1000 3FFF
Page 16	0x1000 4000	0x1000 43FF
Page 17	0x1000 4400	0x1000 47FF
Page 18	0x1000 4800	0x1000 4BFF
Page 19	0x1000 4C00	0x1000 4FFF
Page 20	0x1000 5000	0x1000 53FF
Page 21	0x1000 5400	0x1000 57FF
Page 22	0x1000 5800	0x1000 5BFF



Table 2. SRAM2 organization (continued)

Page number	Start address	End address
Page 23	0x1000 5C00	0x1000 5FFF
Page 24	0x1000 6000	0x1000 63FF
Page 25	0x1000 6400	0x1000 67FF
Page 26	0x1000 6800	0x1000 6BFF
Page 27	0x1000 6C00	0x1000 6FFF
Page 28	0x1000 7000	0x1000 73FF
Page 29	0x1000 7400	0x1000 77FF
Page 30	0x1000 7800	0x1000 7BFF
Page 31	0x1000 7C00	0x1000 7FFF
Page 32	0x1000 8000	0x1000 83FF
Page 33	0x1000 8400	0x1000 87FF
Page 34	0x1000 8800	0x1000 8BFF
Page 35	0x1000 8C00	0x1000 8FFF
Page 36	0x1000 9000	0x1000 93FF
Page 37	0x1000 9400	0x1000 97FF
Page 38	0x1000 9800	0x1000 9BFF
Page 39	0x1000 9C00	0x1000 9FFF
Page 40	0x1000 A000	0x1000 A3FF
Page 41	0x1000 A400	0x1000 A7FF
Page 42	0x1000 A800	0x1000 ABFF
Page 43	0x1000 AC00	0x1000 AFFF
Page 44	0x1000 B000	0x1000 B3FF
Page 45	0x1000 B400	0x1000 B7FF
Page 46	0x1000 B800	0x1000 BBFF
Page 47	0x1000 BC00	0x1000 BFFF
Page 48	0x1000 C000	0x1000 C3FF
Page 49	0x1000 C400	0x1000 C7FF
Page 50	0x1000 C800	0x1000 CBFF
Page 51	0x1000 CC00	0x1000 CFFF
Page 52	0x1000 D000	0x1000 D3FF
Page 53	0x1000 D400	0x1000 D7FF
Page 54	0x1000 D800	0x1000 DBFF
Page 55	0x1000 DC00	0x1000 DFFF
Page 56	0x1000 E000	0x1000 E3FF
Page 57	0x1000 E400	0x1000 E7FF

Table 2. SRAM2 organization (continued)

Page number	Start address	End address
Page 58	0x1000 E800	0x1000 EBFF
Page 59	0x1000 EC00	0x1000 EFFF
Page 60	0x1000 F000	0x1000 F3FF
Page 61	0x1000 F400	0x1000 F7FF
Page 62	0x1000 F800	0x1000 FBFF
Page 63	0x1000 FC00	0x1000 FFFF

The write protection can be enabled in [SYSCFG SRAM2 write protection register \(SYSCFG\\_SWPR\)](#) in the SYSCFG block. This is a register with write '1' once mechanism, which means by writing '1' on a bit it will setup the write protection for that page of SRAM and it can be removed/cleared by a system reset only.

### 2.4.3 SRAM2 Read protection

The SRAM2 is protected with the Read protection (RDP). Refer to [Section 3.5.1: Read protection \(RDP\)](#) for more details.

### 2.4.4 SRAM2 Erase

The SRAM2 can be erased with a system reset using the option bit SRAM2\_RST in the user option byte (refer to [Section 3.4.1: Option bytes description](#)).

The SRAM2 erase can also be requested by software by setting the bit SRAM2ER in the [SYSCFG SRAM2 control and status register \(SYSCFG\\_SCSR\)](#).

## 2.5 Flash memory overview

The Flash memory is composed of two distinct physical areas:

- The main Flash memory block. It contains the application program and user data if necessary.
- The information block. It is composed of three parts:
  - Option bytes for hardware and memory protection user configuration.
  - System memory that contains the ST proprietary code.
  - OTP (one-time programmable) area

The Flash interface implements instruction access and data access based on the AHB protocol. It also implements the logic necessary to carry out the Flash memory operations (program/erase) controlled through the Flash registers Refer to [Section 3: Embedded Flash memory \(FLASH\)](#) for more details.

## 2.6 Boot configuration

### 2.6.1 Boot configuration

Three different boot modes can be selected through the BOOT0 pin or the nBOOT0 bit into the FLASH\_OPTR register (if the nSWBOOT0 bit is cleared into the FLASH\_OPTR register), and nBOOT1 bit in FLASH\_OPTR register, as shown in the following table.

**Table 3. 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	Embedded SRAM1 is selected as boot area
1	X	1	1	X	System memory is selected as boot area
1	0	X	0	X	System memory is selected as boot area

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 pin (coming from the pin or the option bit) and nBOOT1 bit are latched upon reset release. It is up to the user to set nBOOT1 and BOOT0 to select the required boot mode.

The BOOT0 pin or user option bit (depending on the nSWBOOT0 bit value in the FLASH\_OPTR register), and nBOOT1 bit are also re-sampled when exiting from Standby mode. Consequently, they must be kept in the required Boot mode configuration in Standby mode. After this startup delay has elapsed, the CPU fetches the top-of-stack value from address 0x0000 0000, then starts code execution from the boot memory at 0x0000 0004.

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

- Boot from main Flash memory: the main Flash memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x0800 0000). In other words, the Flash memory contents can be accessed starting from address 0x0000 0000 or 0x0800 0000.
- Boot from system memory: the system memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x1FFF 0000).
- Boot from the embedded SRAM1: the SRAM1 is aliased in the boot memory space (0x0000 0000), but it is still accessible from its original memory space (0x2000 0000).

PH3/BOOT0 GPIO is configured in:

- Input mode during the complete reset phase if the option bit nSWBOOT0 is set into the FLASH\_OPTR register and then switches automatically in analog mode after reset is released (BOOT0 pin).
- Input mode from the reset phase to the completion of the option byte loading if the bit nSWBOOT0 is cleared into the FLASH\_OPTR register (BOOT0 value coming from the option bit). It switches then automatically to the analog mode even if the reset phase is not complete.

*Note:* When the device boots from SRAM, in the application initialization code, you have to relocate the vector table in SRAM using the NVIC exception table and the offset register. When booting from the main Flash memory, the application software can either boot from bank 1 or from bank 2. By default, boot from bank 1 is selected. To select boot from Flash memory bank 2, set the BFB2 bit in the user option bytes. When this bit is set and the boot pins are in the boot from main Flash memory configuration, the device boots from system memory, and the boot loader jumps to execute the user application programmed in Flash memory bank 2. For further details, please refer to AN2606.

### Physical remap

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

The following memories can thus be remapped:

- Main Flash memory
- System memory
- Embedded SRAM1 (192 KB)
- FSMC bank 1 (NOR/PSRAM 1 and 2)
- OctoSPI (OCTOSPI1 or OSCTOSPI2) memory

**Table 4. Memory mapping versus boot mode/physical remap**

Addresses	Boot/remap in main Flash memory	Boot/remap in embedded SRAM 1	Boot/remap in system memory	Remap in FSMC	Remap in OCTOSPI
0x2000 0000 - 0x2002 FFFF	SRAM1	SRAM1	SRAM1	SRAM1	SRAM1
0x1FFF 7000 - 0x1FFF FFFF	System memory/OTP/Options bytes	System memory/OTP/Options bytes	System memory/OTP/Options bytes	System memory/OTP/Options bytes	System memory/OTP/Options bytes
0x1000 8000 - 0x1FFE FFFF	Reserved	Reserved	Reserved	Reserved	Reserved
0x1000 0000 - 0x1000 FFFF	SRAM2	SRAM2	SRAM2	SRAM2	SRAM2
0x0820 0000 - 0x0FFF FFFF	Reserved	Reserved	Reserved	Reserved	Reserved
0x0800 0000 - 0x081F FFFF	Flash memory	Flash memory	Flash memory	Flash memory	Flash memory

**Table 4. Memory mapping versus boot mode/physical remap (continued)**

Addresses	Boot/remap in main Flash memory	Boot/remap in embedded SRAM 1	Boot/remap in system memory	Remap in FSMC	Remap in OCTOSPI
0x0400 0000 - 0x07FF FFFF	Reserved	Reserved	Reserved	FSMC bank 1 NOR/PSRAM 2 (128 MB) Aliased	OCTOSPI bank (128 MB) Aliased
0x0010 0000 - 0x03FF FFFF	Reserved	Reserved	Reserved	FSMC bank 1 NOR/PSRAM 1 (128 MB) Aliased	OCTOSPI bank (128 MB) Aliased
0x0000 0000 - 0x001F FFFF (1) (2)	Flash (2 MB) Aliased	SRAM1 (192 KB) Aliased	System memory (28 KB) Aliased	FSMC bank 1 NOR/PSRAM 1 (128 MB) Aliased	OCTOSPI bank (128 MB) Aliased

1. When the FSMC is remapped at address 0x0000 0000, only the first two regions of bank 1 memory controller (bank 1 NOR/PSRAM 1 and NOR/PSRAM 2) can be remapped. When the OCTOSPI is remapped at address 0x0000 0000, only 128 MB are remapped. In remap mode, the CPU can access the external memory via ICode bus instead of system bus, which boosts up the performance.
2. Even when aliased in the boot memory space, the related memory is still accessible at its original memory space.

**Embedded boot loader**

The embedded boot loader is located in the system memory, programmed by ST during production. Refer to AN2606 STM32 microcontroller system memory boot mode.

## 3 Embedded Flash memory (FLASH)

### 3.1 Introduction

The Flash memory interface manages CPU AHB ICode and DCode accesses to the Flash memory. It implements the erase and program Flash memory operations and the read and write protection mechanisms.

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

### 3.2 FLASH main features

- Up to 2 Mbytes of Flash memory with dual-bank architecture supporting read-while-write capability (RWW).
- Flash memory read operations with two data width modes supported:
  - Single-bank mode DBANK=0: read access of 128 bits
  - Dual-bank mode DBANK=1: read access of 64 bits
- Page erase, bank erase and mass erase (both banks)

Flash memory interface features:

- Flash memory read operations
- Flash memory program/erase operations
- Read protection activated by option (RDP)
- 4 Write protection areas (2 per bank when DBANK=1 and 4 for full memory when DBANK=0)
- 2 proprietary code read protection areas (1 per bank when DBANK=1, 2 for all memory when DBANK=0)
- Flash empty check
- Prefetch on ICODE
- Instruction Cache: 32 cache lines of 4 x 64 or 2 x 128 bits on ICode (1 Kbyte RAM)
- Data Cache: 8 cache lines of 4 x 64 bits or 2 x 128 on DCode (256 bytes RAM)
- Error Code Correction ECC: 8 bits per 64-bit double-word
  - DBANK=1: 8 + 64 = 72 bits, 2 bits detection, 1 bit correction
  - DBANK=0: (8+64) + (8+64) = 144 bits, 2 bits detection, 1 bit correction
- Option byte loader
- Low-power mode

## 3.3 FLASH functional description

### 3.3.1 Flash memory organization

The Flash memory has the following main features:

- Capacity up to 2 Mbytes, in Single-bank mode (read width of 128 bits) or in Dual-bank mode (read width of 64-bits)
- Supports dual boot mode thanks to the BFB2 option bit (only in Dual-bank mode)
- Dual-bank mode when DBANK bit is set:
  - 2 Mbytes organized in 2 banks for main memory
  - Page size of 4 Kbytes
  - 72 bits wide data read (64 bits plus 8 ECC bits)
  - Bank and Mass erase
- Single-bank mode when DBANK is reset:
  - 2 Mbytes organized in one single-bank for main memory
  - Page size of 8 Kbytes
  - 144 bits wide data read (64 bits plus 2x8 ECC bits)
  - Mass erase

The Flash memory is organized as follows:

- A main memory block organized depending on the dual-bank configuration bit:
  - When dual-bank is enabled (DBANK bit set), the Flash is divided in 2 banks of 1 Mbyte, and each bank is organized as follows:  
The main memory block containing 256 pages of 4 Kbytes  
Each page is composed of 8 rows of 512 bytes
  - When dual-bank is disabled (DBANK bit reset), the main memory block is organized as one single-bank of 2 Mbytes as follows:  
The main memory block containing 256 pages of 8 Kbytes  
Each page is composed of 8 rows of 1024 bytes
- Dual-bank organization on 1 Mbyte devices  
The dual-bank feature on 1 Mbyte devices is enabled by setting the DB1M option bit. The dual-bank memory organization is different from the single-bank: the single-bank memory contains 256 pages of 8 Kbytes whereas the dual-bank memory each bank contains 128 pages of 4 Kbytes.

For erase operation, the right page numbering and address must be considered according to the DB1M option bit.

- When the DB1M bit is reset, the erase operation must be performed on the Bank 1.
- When the DB1M bit is set, to perform an erase operation on Bank 2, the page number must be programmed (page number from 0 to 127) on Bank 2 at address 0x0801 0000 (same as sequence for page erase in Bank 2 as on 2 Mbytes devices).
- Refer to [Table 7: Flash module - 1 Mbyte dual-bank organization \(64 bits read width\)](#) and [Table 8: Flash module - 1 Mbyte single-bank organization \(128 bits read width\)](#) for details on 1 Mbyte single-bank and 1 Mbyte dual-bank organizations.
- An Information block containing:
  - System memory from which the device boots in System memory boot mode. The area is reserved for use by STMicroelectronics and contains the bootloader that is used to reprogram the Flash memory through one of the following interfaces: USART1, USART2, USART3, USB (DFU), I2C1, I2C2, I2C3, SPI1, SPI2, SPI3. It is programmed by STMicroelectronics when the device is manufactured, and protected against spurious write/erase operations. For further details, please refer to the AN2606 available from [www.st.com](http://www.st.com).
  - 1 Kbyte (128 double word) OTP (one-time programmable) bytes for user data. The OTP area is available in Bank 1 only. The OTP data cannot be erased and can be written only once. If only one bit is at 0, the entire double word 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 the tables below.



**Table 5. Flash module - 2 Mbytes dual-bank organization (64 bits read width)**

Flash area		Flash memory address	Size (bytes)	Name
Main memory	Bank 1	0x0800 0000 - 0x0800 0FFF	4 K	Page 0
		0x0800 1000 - 0x0800 1FFF	4 K	Page 1
		0x0800 2000 - 0x0800 2FFF	4 K	Page 2
		0x0800 3000 - 0x0800 3FFF	4 K	Page 3
		-	-	-
		-	-	-
		-	-	-
	0x080F F000 - 0x080F FFFF	4 K	Page 255	
	Bank 2	0x0810 0000 - 0x0810 0FFF	4 K	Page 0
		0x0810 1000 - 0x0810 1FFF	4 K	Page 1
		0x0810 2000 - 0x0810 2FFF	4 K	Page 2
		0x0810 3000 - 0x0810 3FFF	4 K	Page 3
		-	-	-
		-	-	-
-		-	-	
0x081F F000 - 0x081F FFFF	4 K	Page 255		
Information block	Bank 1	0x1FFF 0000 - 0x1FFF 6FFF	28 K	System memory
	Bank 2	0x1FFF 8000 - 0x1FFF EFFF	28 K	
	Bank 1	0x1FFF 7000 - 0x1FFF 73FF	1 K	OTP area
	Bank 1	0x1FF0 0000 - 0x1FF0 000F	16	Option bytes
	Bank 2	0x1FF0 1000 - 0x1FF0 100F	16	

**Table 6. Flash module - 2 Mbytes single-bank organization (128 bits read width)**

Flash area		Flash memory address	Size (bytes)	Name
Main memory		0x0800 0000 - 0x0800 1FFF	8 K	Page 0
		0x0800 2000 - 0x0800 3FFF	8 K	Page 1
		0x0800 4000 - 0x0800 5FFF	8 K	Page 2
		-	-	-
		-	-	-
		-	-	-
		-	-	-
		0x0801 C000 - 0x0801 DFFF	8 K	Page 244
		0x081F E000 - 0x081F FFFF	8 K	Page 255

**Table 6. Flash module - 2 Mbytes single-bank organization (128 bits read width)**

Flash area		Flash memory address	Size (bytes)	Name
Information block	Bank 1	0x1FFF 0000 - 0x1FFF 6FFF	28 K	System memory
	Bank 2	0x1FFF 8000 - 0x1FFF EFFF	28 K	
	Bank 1	0x1FFF 7000 - 0x1FFF 73FF	1 K	OTP area
	Bank 1	0x1FF0 0000 - 0x1FF0 000F	16	Option bytes
	Bank 2	0x1FF0 1000 - 0x1FF0 100F	16	

**Table 7. Flash module - 1 Mbyte dual-bank organization (64 bits read width)**

Flash area		Flash memory address	Size (bytes)	Name
Main memory 1 Mbyte	Bank 1 512 Kbytes	0x0800 0000 - 0x0800 0FFF	4 K	Page 0
		0x0800 1000 - 0x0800 1FFF	4 K	Page 1
		0x0800 2000 - 0x0800 2FFF	4 K	Page 2
		0x0800 3000 - 0x0800 3FFF	4 K	Page 3
		-	-	-
		-	-	-
		-	-	-
		0x0807 F000 - 0x0807 FFFF	4 K	Page 127
	Bank 2 <sup>(1)</sup> 512 Kbytes	0x0808 0000 - 0x0808 0FFF	4 K	Page 0
		0x0808 1000 - 0x0808 1FFF	4 K	Page 1
		0x0808 2000 - 0x0808 2FFF	4 K	Page 2
		0x0808 3000 - 0x0808 3FFF	4 K	Page 3
		-	-	-
		-	-	-
-		-	-	
	0x080F F000 - 0x080F FFFF	4 K	Page 127	
Information block	Bank 1	0x1FFF 0000 - 0x1FFF 6FFF	28 K	System memory
	Bank 2	0x1FFF 8000 - 0x1FFF EFFF	28 K	
	Bank 1	0x1FFF 7000 - 0x1FFF 73FF	1 K	OTP area
	Bank 1	0x1FF0 0000 - 0x1FF0 000F	16	Option bytes
	Bank 2	0x1FF0 1000 - 0x1FF0 100F	16	

1. Erase operations on Bank 2 must be performed in the same way as on the 2 Mbytes devices. Bank 2 address for erase must be 0x0801 0000.

**Table 8. Flash module - 1 Mbyte single-bank organization (128 bits read width)**

Flash area		Flash memory address	Size (bytes)	Name
Main memory		0x0800 0000 - 0x0800 1FFF	8 K	Page 0
		0x0800 2000 - 0x0800 3FFF	8 K	Page 1
		0x0800 4000 - 0x0800 5FFF	8 K	Page 2
		-	-	-
		-	-	-
		0x080F E000 - 0x080F FFFF	8 K	Page 127
Information block	Bank 1	0x1FFF 0000 - 0x1FFF 6FFF	28 K	System memory
	Bank 2	0x1FFF 8000 - 0x1FFF EFFF	28 K	
	Bank 1	0x1FFF 7000 - 0x1FFF 73FF	1 K	OTP area
	Bank 1	0x1FF0 0000 - 0x1FF0 000F	16	Option bytes
	Bank 2	0x1FF0 1000 - 0x1FF0 100F	16	

### 3.3.2 Error code correction (ECC)

#### Dual-bank mode (DBANK=1, 64-bits data width)

Data in Flash memory are 72-bits words: 8 bits are added per double word (64 bits). The ECC mechanism supports:

- One error detection and correction
- Two errors detection

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

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

When an ECC error is detected, the address of the failing double word and its associated bank are saved in ADDR\_ECC[20:0] and BK\_ECC in the FLASH\_ECCR register. ADDR\_ECC[2:0] are always cleared.

When ECCC or ECCD is set, ADDR\_ECC and BK\_ECC are not updated if a new ECC error occurs. FLASH\_ECCR is updated only when ECC flags are cleared.

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

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

### Single-bank mode (DBANK=0, 128-bits data width)

Data in Flash memory are 144-bits words: 8 bits are added per each double word. The ECC mechanism supports:

- One error detection and correction
- Two errors detection per 64 double words

The user must first check the SYSF\_ECC bit, and if it is set, the user must refer to the DBANK=1 programming model (because system Flash is always on 2 banks). If the bit is not set, the user must refer to the following programming model:

Each double word (bits 63:0 and bits 127:64) has ECC.

When one error is detected in 64 LSB bits (bits 63:0) and corrected, a flag ECCC (ECC correction) is set in the FLASH\_ECCR register.

When one error is detected in 64 MSB bits (bits 127:64) and corrected, a flag ECCC2 (ECC2 correction) is set in the FLASH\_ECCR register.

If the ECCIE is set, an interrupt is generated. The user has to read ECCC and ECCC2 to see which part of the 128-bits data has been corrected (either 63:0, 127:64 or both).

When two errors are detected in 64 LSB bits, a flag ECCD (ECC detection) is set in the FLASH\_ECCR register.

When two errors are detected in 64 MSB bits (bits 127:64), a flag ECCD2 (ECC2 detection) is set in the FLASH\_ECCR register.

In this case, a NMI is generated. The user has to read ECCD and ECCD2 to see which part of the 128-bits data has error detection (either 63:0, 127:64 or both).

When an ECC error is detected, the address of the failing the 2 times double word is saved into ADDR\_ECC[20:0] in FLASH\_ECCR. ADDR\_ECC[20:0] contains an address of a 2 times double word.

The ADDR\_ECC[3:0] are always cleared. BK\_ECC is not used in this mode.

When ECCC/ECCC2 or ECCD/ECCD2 is/are set, if a new ECC error occurs, the ADDR\_ECC is not updated. The FLASH\_ECCR is updated only if the ECC flags (ECCC/ECCC2/ECCD/ECCD2) 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.*

### 3.3.3 Read access latency

To correctly read data from Flash memory, the number of wait states (LATENCY) must be correctly programmed in the *Flash access control register (FLASH\_ACR)* according to the frequency of the CPU clock (HCLK) and the internal voltage range of the device  $V_{CORE}$ . Refer to *Section 5.1.7: Dynamic voltage scaling management. Table 9* shows the correspondence between wait states and CPU clock frequency.

**Table 9. Number of wait states according to CPU clock (HCLK) frequency**

Wait states (WS) (Latency)	HCLK (MHz)	
	V <sub>CORE</sub> Range 1	V <sub>CORE</sub> Range 2
0 WS (1 CPU cycles)	≤20	≤8
1 WS (2 CPU cycles)	≤40	≤16
2 WS (3 CPU cycles)	≤60	≤26
3 WS (4 CPU cycles)	≤80	-
4 WS (5 CPU cycles)	≤100	-
5 WS (6 CPU cycles)	≤120	-

After reset, the CPU clock frequency is 4 MHz and 0 wait state (WS) is configured in the FLASH\_ACR register.

When changing the CPU frequency, the following software sequences must be applied in order to tune the number of wait states needed to access the Flash memory:

#### Increasing the CPU frequency:

1. Program the new number of wait states to the LATENCY bits in the *Flash access control register (FLASH\_ACR)*.
2. Check that the new number of wait states is taken into account to access the Flash memory by reading the FLASH\_ACR register.
3. Modify the CPU clock source by writing the SW bits in the RCC\_CFGR register.
4. If needed, modify the CPU clock prescaler by writing the HPRE bits in RCC\_CFGR.
5. Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in the RCC\_CFGR register.

#### Decreasing the CPU frequency:

1. Modify the CPU clock source by writing the SW bits in the RCC\_CFGR register.
2. If needed, modify the CPU clock prescaler by writing the HPRE bits in RCC\_CFGR.
3. Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in the RCC\_CFGR register.
4. Program the new number of wait states to the LATENCY bits in *Flash access control register (FLASH\_ACR)*.
5. Check that the new number of wait states is used to access the Flash memory by reading the FLASH\_ACR register.

### 3.3.4 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 requires 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 which 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 120 MHz.

### Instruction prefetch

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

In case of Single-bank mode (DBANK option bit is reset), each Flash memory read operation provides 128 bits from either four instructions of 32 bits or eight instructions of 16 bits depending on the launched program. This 128-bits current instruction line is saved in a current buffer, and in case of sequential code, at least four CPU cycles are needed to execute the previous read instruction line.

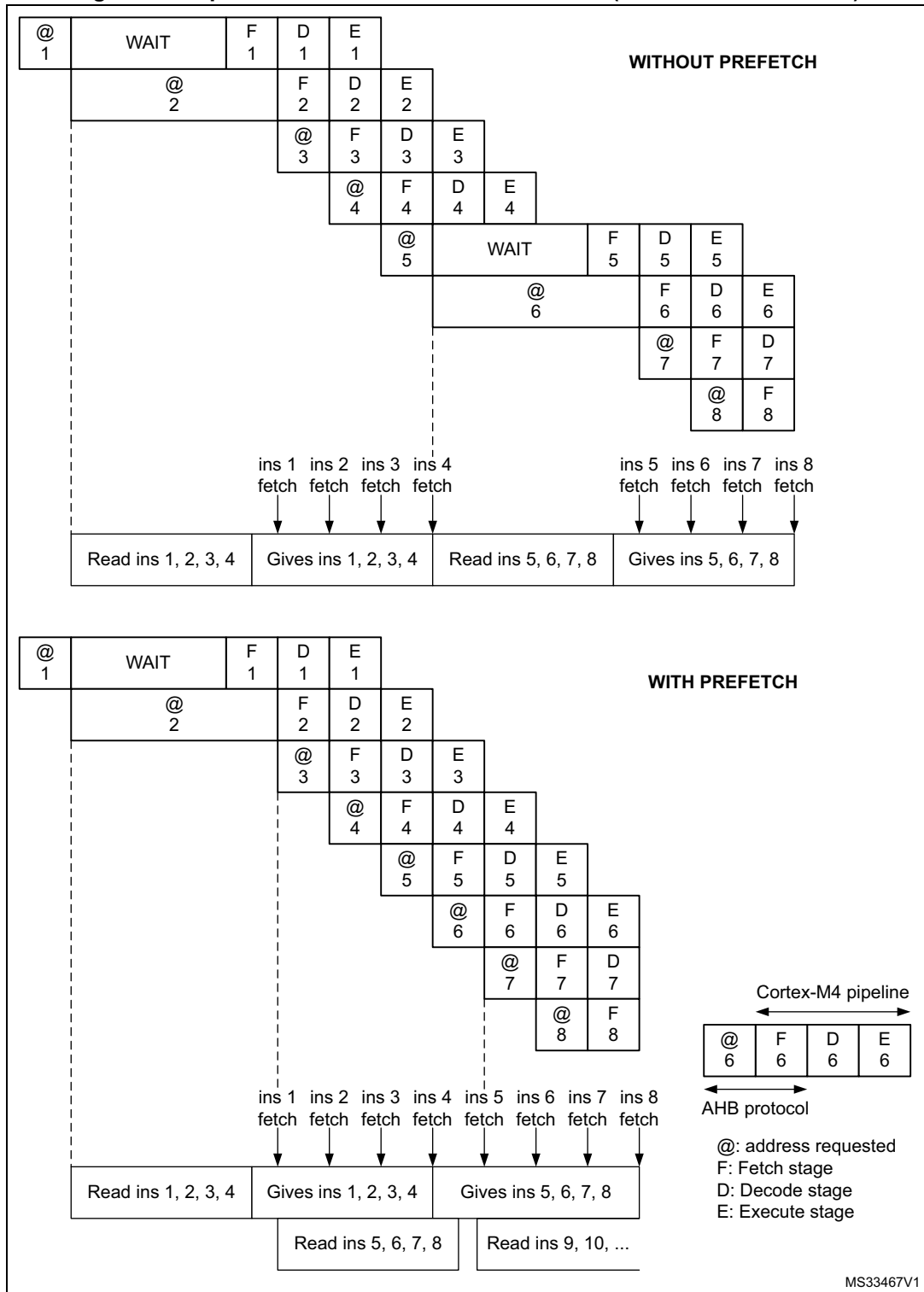
When in Dual-bank mode (DBANK option bit is set), each Flash memory read operation provides 64 bits from either two instructions of 32 bits or four instructions of 16 bits depending on the launched program. This 64-bits current instruction line is saved in a current buffer, and in case of sequential code, at least two CPU cycles are needed to execute the previous read instruction line.

Prefetch on the ICode 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 access control register (FLASH\_ACR)*. 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 3 WS are needed to access the Flash memory.

Figure 3. Sequential 16-bit instructions execution (64-bit read data width)



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 Flash access is performed.

### Instruction cache memory (I-Cache)

To limit the time lost due to jumps, it is possible to retain 32 lines of 4 x 64 bits in Dual-bank mode or 32 lines of 2 x 128 bits in Single-bank mode in an instruction cache memory. This feature can be enabled by setting the instruction cache enable (ICEN) bit in the *Flash access control register (FLASH\_ACR)*. Each time a miss occurs (requested data not present in the currently used instruction line, in the prefetched instruction line or in the instruction cache memory), the line read is copied into the instruction cache memory. If some data contained in the instruction cache memory are requested by the CPU, they are provided without inserting any delay. Once all the instruction cache memory lines 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.

### Data cache memory (D-Cache)

Literal pools are fetched from Flash memory through the DCode bus during the execution stage of the CPU pipeline. Each DCode bus read access fetches 64 or 128 bits which 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 data cache memory can be enabled by setting the data cache enable (DCEN) bit in the *Flash access control register (FLASH\_ACR)*. This feature works like the instruction cache memory, but the retained data size is limited to 8 rows of 4x64 bits in Dual-bank mode and to 8 rows of 2x128 bits in Single-bank mode.

The Data cache memory is enable after system reset.

*Note: The D-Cache is active only when data is requested by the CPU (not by DMA1 and DMA2). Data in option bytes block are not cacheable.*

## 3.3.5 Flash program and erase operations

The STM32L4Rxxx and STM32L4Sxxx embedded Flash memory can be programmed using in-circuit programming or in-application programming.

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

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

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

An on-going Flash memory operation will not block the CPU as long as the CPU does not access the same Flash memory bank. Code or data fetches are possible on one bank while



a write/erase operation is performed to the other bank (refer to [Section 3.3.8: Read-while-write \(RWW\) available only in Dual-bank mode \(DBANK=1\)](#)). The Flash erase and programming is only possible in the voltage scaling range 1. The VOS[1:0] bits in the PWR\_CR1 must be programmed to 01b.

On the contrary, during a program/erase operation to the Flash memory, any attempt to read the same Flash memory bank will stall the bus. The read operation will proceed correctly once the program/erase operation has completed.

### Unlocking the Flash memory

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

1. Write KEY1 = 0x45670123 in the [Flash key register \(FLASH\\_KEYR\)](#)
2. Write KEY2 = 0xCDEF89AB in the FLASH\_KEYR register.

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

The FLASH\_CR register can be locked again by software by setting the LOCK bit in the FLASH\_CR register.

*Note:* The FLASH\_CR register cannot be written when the BSY bit in the [Flash status register \(FLASH\\_SR\)](#) is set. Any attempt to write to it with the BSY bit set will cause the AHB bus to stall until the BSY bit is cleared.

### 3.3.6 Flash main memory erase sequences

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

#### Page erase

To erase a page, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the [Flash status register \(FLASH\\_SR\)](#).
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. In Dual-bank mode (DBANK option bit is set), set the PER bit and select the page to erase (PNB) with the associated bank (BKER) in the Flash control register (FLASH\_CR). In Single-bank mode (DBANK option bit is reset), set the PER bit and select the page to erase (PNB). The BKER bit in the Flash control register (FLASH\_CR) must be kept cleared.
4. Set the STRT bit in the FLASH\_CR register.
5. Wait for the BSY bit to be cleared in the FLASH\_SR 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.

If the page erase is part of write-protected area (by WRP or PCROP), WRPER is set and the page erase request is aborted.

### Bank 1, Bank 2 Mass erase (available only in Dual-bank mode when DBANK=1)

To perform a bank Mass Erase, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH\_SR register.
2. Check and clear all error programming flags due to a previous programming. If not, PGSEERR is set.
3. Set the MER1 bit or MER2 (depending on the bank) in the *Flash control register (FLASH\_CR)*. Both banks can be selected in the same operation, in that case it corresponds to a mass erase.
4. Set the STRT bit in the FLASH\_CR register.
5. Wait for the BSY bit to be cleared in the *Flash status register (FLASH\_SR)*.

#### Mass erase

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\_SR register.
2. Check and clear all error programming flags due to a previous programming. If not, PGSEERR is set.
3. Set the MER1 bit and MER2 in the Flash control register (FLASH\_CR).
4. Set the STRT bit in the FLASH\_CR register.
5. Wait for the BSY bit to be cleared in the Flash status register (FLASH\_SR).

*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.

*When DBANK=0, if only the MERA or the MERB bit is set, PGSEERR is set and no erase operation is performed.*

*If the bank to erase or if one of the banks to erase contains a write-protected area (by WRP or PCROP), WRPEERR is set and the mass erase request is aborted (for both banks if both are selected).*

### 3.3.7 Flash main memory programming sequences

The Flash memory is programmed 72 bits at a time (64 bits + 8 bits ECC).

Programming in a previously programmed address is not allowed except if the data to write is full zero, and any attempt will set PROGERR flag in the *Flash status register (FLASH\_SR)*.

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

- Any attempt to write byte or half-word will set SIZERR flag in the FLASH\_SR register.
- Any attempt to write a double word which is not aligned with a double word address will set PGAERR flag in the FLASH\_SR register.

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

*Note:* When the Flash 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 for a row (64 double words if DBANK=1) or for half row (64 double words if DBANK=0)**

This mode allows to program a row (64 double words if DBANK=1) or half row (64 double words if DBANK=0), and to reduce the page programming time by eliminating the need for verifying the Flash locations before they are programmed and to avoid rising and falling time of high voltage for each double word. During fast programming, the CPU clock frequency (HCLK) must be at least 8 MHz.

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

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

1. In Single-bank mode (DBANK=0), perform a mass erase. If not, PGSERR is set. The Fast programming can be performed only if the code is executed from RAM or from

- bootloader. In Dual-bank mode (DBANK=1), perform a mass erase of the bank to program. If not, PGSERR is set.
2. Check that no Flash main memory operation is ongoing by checking the BSY bit in the *Flash status register (FLASH\_SR)*.
  3. Check and clear all error programming flag due to a previous programming.
  4. Set the FSTPG bit in *Flash control register (FLASH\_CR)*.
  5. Write the 64 double words to program a row or half row. Only double words can be programmed:
    - Write a first word in an address aligned with double word
    - Write the second word.
  6. Wait until the BSY bit is cleared in the FLASH\_SR register.
  7. Check that EOP flag is set in the FLASH\_SR register (meaning that the programming operation has succeed), and clear it by software.
  8. Clear the FSTPG bit in the FLASH\_SR register if there no more programming request anymore.

**Note:** *If the Flash is attempted to be written in fast programming mode while a read operation is ongoing in the same bank, the programming is aborted without any system notification (no error flag is set).*

*When the Flash 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 for all the programming (around 2 x 25 us). Maximum time between two double words write requests is the time programming. If a second double word arrives after this time programming, fast programming is interrupted and MISSERR is set.*

*High voltage mustn't exceed 8 ms for a full row between 2 erases. This is guaranteed by the sequence of 64 double words successively written with a clock system greater or equal to 8MHz. An internal time-out counter counts 7ms when fast programming is set and stops the programming when time-out is over. In this case the FASTERR bit is set.*

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

## Programming errors

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

- **PROGERR:** Programming error  
In standard programming: PROGERR is set if the word to write is not previously erased (except if the value to program is full zero).
- **SIZERR:** Size programming error  
In standard programming or in fast programming: only double word can be programmed and only 32-bit data can be written. SIZERR is set if a byte or an half-word is written.
- **PGAERR:** Alignment programming error  
PGAERR is set if one of the following conditions occurs:
  - In standard programming: the first word to be programmed is not aligned with a double word address, or the second word doesn't belong to the same double word address.
  - In fast programming: the data to program 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: a data is written when PG and FSTPG are cleared.
  - In the standard programming sequence or the fast programming sequence: MER1, MER2, and PER are not cleared when PG or FSTPG is set.
  - In the fast programming sequence: the Mass erase is not performed before setting FSTPG bit.
  - In the mass erase sequence: PG, FSTPG, and PER are not cleared when MER1 or MER2 is set.
  - In the page erase sequence: PG, FSTPG, MER1 and MER2 are not cleared when PER is set.
  - PGSERR is set also if PROGERR, SIZERR, PGAERR, WRPERR, MISSERR, FASTERR or PGSERR is set due to a previous programming error.
  - When DBANK=0, in the case that only either MER1 or MER2 is set, PGSERR is set (bank mass erase is not allowed).
- **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 bank 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.
- **MISSERR:** Fast programming data miss error  
In fast programming: all the data must be written successively. MISSERR is set if the

previous data programming is finished and the next data to program is not written yet.

- **FASTERR**: Fast programming error

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

- When FSTPG bit is set for more than 7ms which generates a time-out detection.
- When the 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\_SR register:

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 status register (FLASH\_SR)*, an interrupt is generated and the operation error flag OPERR is set in the FLASH\_SR 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 requests.

### Programming and caches

If a Flash memory write access concerns some data in the data cache, the Flash write access modifies the data in the Flash memory and the data in the cache.

If an erase operation in Flash memory also concerns data in the data or instruction cache, you have to make sure that these data are rewritten before they are accessed during code execution. If this cannot be done safely, it is recommended to flush the caches by setting the DCRST and ICRST bits in the *Flash access control register (FLASH\_ACR)*.

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

### 3.3.8 Read-while-write (RWW) available only in Dual-bank mode (DBANK=1)

The Dual-bank mode is available only when the DBANK option bit is reset, allowing read-while-write operations. This feature allows to perform a read operation from one bank while erase or program operation is performed to the other bank.

*Note:* Write-while-write operations are not allowed. As an example, It is not possible to perform an erase operation on one bank while programming the other one.

#### Read from bank 1 while page erasing in bank 2 (or vice versa)

While executing a program code from bank 1, it is possible to perform a page erase operation on bank 2 (and vice versa). Follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the *Flash status register (FLASH\_SR)* (BSY is active when erase/program operation is on going in bank 1 or bank 2).
2. Set PER bit, PSB to select the page and BKER to select the bank in the *Flash control register (FLASH\_CR)*.
3. Set the STRT bit in the FLASH\_CR register.
4. Wait for the BSY bit to be cleared (or use the EOP interrupt).

**Read from bank 1 while mass erasing bank 2 (or vice versa)**

While executing a program code from bank 1, it is possible to perform a mass erase operation on bank 2 (and vice versa). Follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the *Flash status register (FLASH\_SR)* (BSY is active when erase/program operation is on going in bank 1 or bank 2).
2. Set MER1 or MER2 to in the *Flash control register (FLASH\_CR)*.
3. Set the STRT bit in the FLASH\_CR register.
4. Wait for the BSY bit to be cleared (or use the EOP interrupt).

**Read from bank 1 while programming bank 2 (or vice versa)**

While executing a program code from bank 1, it is possible to perform a program operation on the bank 2. (and vice versa). Follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the *Flash status register (FLASH\_SR)* (BSY is active when erase/program operation is on going on bank 1 or bank 2).
2. Set the PG bit in the *Flash control register (FLASH\_CR)*.
3. Perform the data write operations at the desired address memory inside the main memory block or OTP area.
4. Wait for the BSY bit to be cleared (or use the EOP interrupt).

### 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 as follows in the option bytes:

**Table 10. Option byte format**

63-24	23-16	15 -8	7-0	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 inside the information block is as shown in [Table 11](#).

The option bytes can be read from the memory locations listed in [Table 11](#) or from the Option byte registers:

- [Flash option register \(FLASH\\_OPTR\)](#)
- [Flash PCROP1 Start address register \(FLASH\\_PCROP1SR\)](#)
- [Flash PCROP1 End address register \(FLASH\\_PCROP1ER\)](#)
- [Flash WRP1 area A address register \(FLASH\\_WRP1AR\)](#)
- [Flash WRP1 area B address register \(FLASH\\_WRP1BR\)](#)
- [Flash PCROP2 Start address register \(FLASH\\_PCROP2SR\)](#)
- [Flash PCROP2 End address register \(FLASH\\_PCROP2ER\)](#)
- [Flash WRP2 area A address register \(FLASH\\_WRP2AR\)](#)
- [Flash WRP2 area B address register \(FLASH\\_WRP2BR\)](#).

**Table 11. Option byte organization**

BANK	Address	[63:56]	[55:48]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]
Bank 1	1FF00000	USER OPT			RDP	USER OPT			RDP
	1FF00008	Unused and PCROP1_STRT[16]		PCROP1_STRT[15:0]		Unused and PCROP1_STRT[16]		PCROP1_STRT[15]	
	1FF00010	PCROP_RDP and Unused and PCROP1_END[16]		PCROP1_END[15:0]		PCROP_RDP and Unused and PCROP1_END[16]		PCROP1_END[15:0]	
	1FF00018	Unused	WRP1A_END	Unused	WRP1A_STRT	Unused	WRP1A_END	Unused	WRP1A_STRT
	1FF00020	Unused	WRP2A_END	Unused	WRP2A_STRT	Unused	WRP2A_END	Unused	WRP2A_STRT



Table 11. Option byte organization (continued)

BANK	Address	[63:56]	[55:48]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]
Bank 2	1FF01000	Unused							
	1FF01008	Unused and PCROP2_STRT[16]		PCROP2_STRT[15]		Unused and PCROP2_STRT[16]		PCROP2_STRT[15]	
	1FF01010	Unused and PCROP2_END[16]		PCROP2_END[15]		Unused and PCROP2_END[16]		PCROP2_END[15]	
	1FF01018	Unused	WRP1A_END	Unused	WRP1B_STRT	Unused	WRP1A_END	Unused	WRP1B_STRT
	1FF01020	Unused	WRP2A_END	Unused	WRP2B_STRT	Unused	WRP2A_END	Unused	WRP2B_STRT

User and read protection option bytes

Flash memory address: 0x1FF0 0000

ST production value: 0xFFEF F8AA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	nBOOT0	nSWBOOT0	SRAM2_RST	SRAM2_PE	nBOOT1	DBANK	DB1M	BFB2	WWDG_SW	IWGD_STDBY	IWDG_STOP	IWDG_SW
				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	Res.	BOR_LEV[2:0]			RDP[7:0]							
	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 **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 Erase when system reset

0: SRAM2 erased when a system reset occurs

1: SRAM2 is 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, this bit selects boot mode from the Flash main memory, SRAM1 or the System memory. Refer to [Section 2.6: Boot configuration](#).

- Bit 22 **DBANK:**
  - 0: Single-bank mode with 128 bits data read width
  - 1: Dual-bank mode with 64 bits data

This bit can be written only when PCROP1/2 is disabled.
- Bit 21 **DB1M:** Dual-bank on 1 Mbyte Flash memory devices
  - 0: 1Mbyte single Flash contiguous address in bank1
  - 1: 1Mbyte Dual-bank Flash with contiguous addresses
- Bit 20 **BFB2:** Dual-bank boot
  - 0: Dual-bank boot disable
  - 1: Dual-bank boot enable
- Bit 19 **WWDG\_SW:** Window watchdog selection
  - 0: Hardware window watchdog
  - 1: Software window watchdog
- Bit 18 **IWDG\_STDBY:** Independent watchdog counter freeze in Standby mode
  - 0: Independent watchdog counter is frozen in Standby mode
  - 1: Independent watchdog counter is running in Standby mode
- Bit 17 **IWDG\_STOP:** Independent watchdog counter freeze in Stop mode
  - 0: Independent watchdog counter is frozen in Stop mode
  - 1: Independent watchdog counter is running in Stop mode
- Bit 16 **IDWG\_SW:** Independent watchdog selection
  - 0: Hardware independent watchdog
  - 1: Software independent watchdog
- Bit 15 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 generate 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
- Bit 11 Reserved, must be kept at reset value.
- Bits10:8 **BOR\_LEV:** BOR reset Level
 

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

  - 000: BOR Level 0. Reset level threshold is around 1.7 V
  - 001: BOR Level 1. Reset level threshold is around 2.0 V
  - 010: BOR Level 2. Reset level threshold is around 2.2 V
  - 011: BOR Level 3. Reset level threshold is around 2.5 V
  - 100: BOR Level 4. Reset level threshold is around 2.8 V
- Bits 7:0 **RDP:** Read protection level
  - 0xAA: Level 0, read protection not active
  - 0xCC: Level 2, chip read protection active
  - Others: Level 1, memories read protection active

**PCROP1 Start address option bytes**

Flash memory address: 0x1FF0 0008

ST production value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1_STRT [16:0]
															r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCROP1_STRT[16:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 16:0 **PCROP1\_STRT[16:0]**: PCROP area start offset

DBANK=1

PCROP1\_STRT contains the first double-word of the PCROP area for bank1.

DBANK=0

PCROP1\_STRT contains the first 2xdouble-word of the PCROP area for all memory.

**PCROP1 End address option bytes**

Flash memory address: 0x1FF0 0010

ST production value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCROP_RDP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1_END
r/w															r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCROP1_END[16:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bit 31 **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 30:17 Reserved, must be kept at reset value.

Bits 16:0 **PCROP1\_END**: Bank 1 PCROP area end offset

DBANK=1

PCROP1\_END contains the last double-word of the bank 1 PCROP area.

DBANK=0

PCROP1\_END contains the last 2x double-word PCROP area for all memory.



**WRP Area A address option bytes**

Flash memory address: 0x1FF0 0018

ST production value: 0x0000 00FF

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

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

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

DBANK=1

WRP1A\_END contains the last page of WRP first area in bank1.

DBANK=0

WRP1A\_END contains the last page of WRP first area for all memory.

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

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

DBANK=1

WRP1A\_STRT contains the first page of WRP first area for bank1.

DBANK=0

WRP1A\_STRT contains the first page of WRP first area for all memory.

**WRP1 Area B address option bytes**

Flash memory address: 0x1FF0 0020

ST production value: 0x0000 00FF

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

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

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

DBANK=1

WRP1B\_END contains the last page of the WRP second area for bank1.

DBANK=0

WRP1B\_END contains the last page of the WPR second area for all memory.

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

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

DBANK=1

WRP1B\_STRT contains the last page of the WRP second area for bank1.

DBANK=0

WRP1B\_STRT contains the last page of the WPR second area for all memory.

### PCROP2 Start address option bytes

Flash memory address: 0x1FF0 1008

ST production value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP2_STRT [16:0]
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCROP2_STRT[16:0]															
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 16:0 **PCROP2\_STRT[16:0]**: PCROP area start offset

DBANK=1

PCROP2\_STRT contains the first double-word of the PCROP area for bank 2.

DBANK=0

PCROP2\_STRT contains the first double-word PCROP area for all memory.

### PCROP2 End address option bytes

Flash memory address: 0x1FF0 1010

ST production value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP2_END [16:0]
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCROP2_END[16:0]															
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 16:0 **PCROP2\_END[16:0]**: PCROP area end offset

DBANK=1

PCROP2\_END contains the last double-word of the PCROP area for bank2.

DBANK=0

PCROP2\_END contains the last 2xdouble-word of the PCROP area for all the memory.

**WRP2 Area A address option bytes**

Flash memory address: 0x1FF0 1018

ST production value: 0x0000 00FF

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

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

Bits 23:16 **WRP2A\_END**: WRP first area “A” end offset

DBANK=1

WRP2A\_END contains the last page of the WRP first area for bank2.

DBANK=0

WRP2A\_END contains the last page of the WRP third area for all memory.

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

Bits 7:0 **WRP2A\_STRT**: WRP first area “A” start offset

DBANK=1

WRP2A\_STRT contains the first page of the WRP first area for bank2.

DBANK=0

WRP2A\_STRT contains the first page of the WRP third area for all memory.

**WRP Area B address option bytes**

Flash memory address: 0x1FF0 1020

ST production value: 0x0000 00FF

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



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

Bits 23:16 **WRP2B\_END**: WRP second area "B" end offset

DBANK=1

WRP2B\_END contains the last page of the WRP second area for bank2.

DBANK=0

WRP2B\_END contains the last page of the WRP fourth area for all memory.

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

Bits 7:0 **WRP2B\_STRT**: WRP second area "B" start offset

DBANK=1

WRP2B\_STRT contains the first page of the WRP second area for bank2.

DBANK=0

WRP2B\_STRT contains the first page of the WRP second area for all memory.

### 3.4.2 Option bytes programming

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

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

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 too.

#### Modifying user options

The option bytes are programmed differently from a main memory user address. It is not possible to modify independently user options of bank 1 or bank 2. The users Options of the bank 1 are modified first.

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

1. Check that no Flash memory operation is on going by checking the BSY bit in the *Flash status register (FLASH\_SR)*.
2. Clear OPTLOCK option lock bit with the clearing sequence described above.
3. Write the desired options value in the options registers: *Flash option register (FLASH\_OPTR)*, *Flash PCROP1 Start address register (FLASH\_PCROP1SR)*, *Flash PCROP1 End address register (FLASH\_PCROP1ER)*, *Flash WRP1 area A address register (FLASH\_WRP1AR)*, *Flash WRP1 area B address register (FLASH\_WRP1BR)*, *Flash PCROP2 Start address register (FLASH\_PCROP2SR)*, *Flash PCROP2 End address register (FLASH\_PCROP2ER)*, *Flash WRP2 area A address register (FLASH\_WRP2AR)*, *Flash WRP2 area B address register (FLASH\_WRP2BR)*.
4. Set the Options Start bit OPTSTRT in the *Flash 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 both user option bytes pages first (bank 1 and bank 2) and then programming all the option bytes with the values contained in the Flash option registers.

### Option byte loading

After the BSY bit is cleared, all new options are updated into the Flash but they are not applied to the system. They will have effect on the system when they are loaded. Option bytes loading (OBL) is performed in two cases:

- when OBL\_LAUNCH bit is set in the *Flash 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 cannot be read with 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 with ECC. 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 which 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"

On system reset rising, internal option registers are copied into option registers which can be read and written by software (FLASH\_OPTR, FLASH\_PCROP1/2SR, FLASH\_PCROP1/2ER, FLASH\_WRP1/2AR, FLASH\_WRP1/2BR). These registers are also used to modify options. If these registers are not modified by user, they reflects the options states of the system. See [Section : Modifying user options](#) for more details.

### Activating Dual-bank mode (switching from DBANK=0 to DBANK=1)

When switching from one Flash mode to another (for example from single to dual-bank) it is recommended to execute the code from the SRAM or use the bootloader. To avoid reading



corrupted data from the Flash when the memory organization is changed, any access (either CPU or DMAs) to Flash memory should be avoided before reprogramming.

- Disable Instruction/data caches and/or prefetch if they are enabled (reset PRFTEN and ICEN/DCEN bits in the FLASH\_ACR register).
- Flush instruction and data cache by setting the DCRST/ICRST bits in the FLASH\_ACR register.
- Set the DBANK option bit and clear all the WRP write protection (follow user option modification and option bytes loader procedure).
  - Once OBL is done with DBANK=0, perform a mass erase.
  - Start a new programming of code in 64 bits mode with DBANK=0 memory mapping.
  - Set the new WRP/PCROP with DBANK=0 scheme if needed.
  - Set PRFTEN and ICEN/DCEN if needed.

The new software is ready to be run using the bank configuration.

### **De-activating Dual-bank mode (switching from DBANK=1 to DBANK=0)**

When switching from one Flash mode to another (for example from single to dual-bank) it is recommended to execute the code from the SRAM or use the bootloader. To avoid reading corrupted data from the Flash when the memory organization is changed, any access (either CPU or DMAs) to Flash memory should be avoided before reprogramming.

- Disable Instruction/data caches and/or prefetch if they are enabled (reset PRFTEN and ICEN/DCEN bits in the FLASH\_ACR register).
- Flush instruction and data cache by setting the DCRST/ICRST bits in the FLASH\_ACR register.
- Clear the DBANK option bit and all WRP write protection (follow user option modification and option bytes loader procedure).
  - Once OBL is done with DBANK=0, perform a mass erase.
  - Start a new programming of code in 128 bits mode with DBANK=0 memory mapping
  - Set the new WRP/PCROP with DBANK=0 scheme if needed. Set PRFTEN and ICEN/DCEN if needed.

The new software is ready to be run using the bank configuration.

### 3.5 FLASH memory protection

The Flash main memory can be protected against external accesses with the Read protection (RDP). The pages of the Flash memory can also be protected against unwanted write due to loss of program counter contexts. The write-protection (WRP) granularity is one page (2 KByte). Apart of the Flash memory can also be protected against read and write from third parties (PCROP). The PCROP granularity is double word (64-bit).

#### 3.5.1 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 to 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 bootloader modes:** In debug mode or when code is running from boot RAM or bootloader, 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 generates a bus error and a Hard Fault interrupt.

**Caution:** In case the Level 1 is configured and no PCROP area is 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 Cortex<sup>®</sup>-M4 debug port, the boot from RAM (boot RAM mode) and the boot from System memory (bootloader mode) are no more available. In user execution mode (boot FLASH mode), all operations are allowed on the Flash Main memory. On the contrary, only read operations can be performed on the option bytes.

Option bytes cannot be programmed nor erased. Thus, the level 2 cannot be removed at all: it is an irreversible operation. When attempting to modify the options bytes, the protection error flag WRPERR is set in the FLASH\_SR register and an interrupt can be generated.

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

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

### Changing the Read protection level

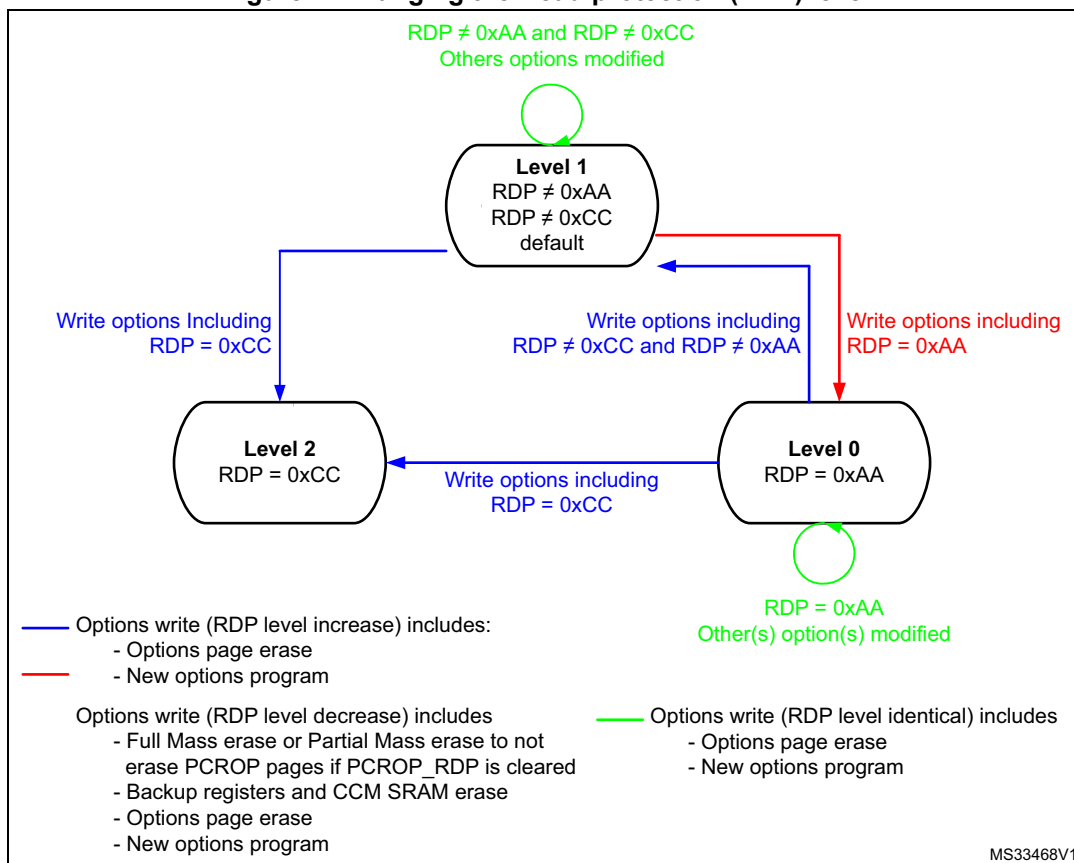
It is easy to move from level 0 to level 1 by changing the value of the RDP byte to any value (except 0xCC). By programming the 0xCC value in the RDP byte, it is possible to go to level 2 either directly from level 0 or from level 1. Once in level 2, it is no more possible to modify the Read protection 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 PCROP1 End address register \(FLASH\\_PCROP1ER\)](#). The backup registers (RTC\_BKPxR in the RTC) and the SRAM2 are also erased. The user options except PCROP protection are set to their previous values copied from FLASH\_OPTR, FLASH\_WRPxyR (x=1, 2 and y=A or B). PCROP is disable. The OTP area is not affected by mass erase and remains unchanged.

If the bit PCROP\_RDP is cleared in the FLASH\_PCROP1ER, the full mass erase is replaced by a partial mass erase that is successive page erases in the bank where PCROP is active, except for the pages protected by PCROP. This is done in order to keep the PCROP code. If PCROP is active for both banks, both banks are erased by page erases. Only when both banks are erased, options are re-programmed with their previous values. This is also true for FLASH\_PCROPxSR and FLASH\_PCROPxER registers (x=1,2).

*Note: Full Mass Erase or 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 control register.*

**Figure 4. Changing the Read protection (RDP) level**



**Table 13. Access status versus protection level and execution modes**

Area	Protection level	User execution (BootFromFlash)			Debug/ BootFromRam/ BootFromLoader <sup>(1)</sup>		
		Read	Write	Erase	Read	Write	Erase
Flash main memory	1	Yes	Yes	Yes	No	No	No <sup>(3)</sup>
	2	Yes	Yes	Yes	N/A	N/A	N/A
System memory <sup>(2)</sup>	1	Yes	No	No	Yes	No	No
	2	Yes	No	No	N/A	N/A	N/A
Option bytes	1	Yes	Yes <sup>(3)</sup>	Yes	Yes	Yes <sup>(3)</sup>	Yes
	2	Yes	No	No	N/A	N/A	N/A

Table 13. Access status versus protection level and execution modes (continued)

Area	Protection level	User execution (BootFromFlash)			Debug/ BootFromRam/ BootFromLoader <sup>(1)</sup>		
		Read	Write	Erase	Read	Write	Erase
OTP	1	Yes	Yes <sup>(4)</sup>	N/A	No	No	N/A
	2	Yes	Yes <sup>(4)</sup>	N/A	N/A	N/A	N/A
Backup registers	1	Yes	Yes	N/A	No	No	No <sup>(5)</sup>
	2	Yes	Yes	N/A	N/A	N/A	N/A
SRAM2	1	Yes	Yes	N/A	No	No	No <sup>(6)</sup>
	2	Yes	Yes	N/A	N/A	N/A	N/A

1. When the protection level 2 is active, the Debug port, the boot from RAM and the boot from system memory are disabled.
2. The system memory is only read-accessible, whatever the protection level (0, 1 or 2) and execution mode.
3. The Flash main memory is erased when the RDP option byte is programmed with all level protections disabled (0xAA).
4. OTP can only be written once.
5. The backup registers are erased when RDP changes from level 1 to level 0.
6. The SRAM2 is erased when RDP changes from level 1 to level 0.

### 3.5.2 Proprietary code readout protection (PCROP)

Apart 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 CPU, as an instruction code, while all other accesses (DMA, debug and CPU data read, write and erase) are strictly prohibited. Depending of the DBANK mode, it allows either to specify one PCROP zone per bank in Dual-bank mode or to specify two PCROP zones for all memory. An additional option bit (PCROP\_RDP) allows to select if the PCROP area is erased or not when the RDP protection is changed from Level 1 to Level 0 (refer to [Changing the Read protection level](#)).

Each PCROP area is defined by a start page offset and an end page offset related to the physical Flash bank base address. These offsets are defined in the PCROP address registers [Flash PCROP1 Start address register \(FLASH\\_PCROP1SR\)](#), [Flash PCROP1 End address register \(FLASH\\_PCROP1ER\)](#), [Flash PCROP2 Start address register \(FLASH\\_PCROP2SR\)](#), [Flash PCROP2 End address register \(FLASH\\_PCROP2ER\)](#).

In Single-bank mode (DBANK=0):

- The PCROP<sub>x</sub> (x = 1,2) area is defined from the address: base address + [PCROP<sub>x</sub>\_STRT x 16] (included) to the address: base address + [(PCROP<sub>x</sub>\_END+1) x 16] (excluded). The minimum PCROP area size is two 2 x double-words (256 bits).

In Dual-bank mode (DBANK=1)

- The PCROP<sub>x</sub> (x = 1,2) area is defined from the address: bank “x” base address + [PCROP<sub>x</sub>\_STRT x 0x8] (included) to the address: bank “x” base address + [(PCROP<sub>x</sub>\_END+1) x 0x8] (excluded). The minimum PCROP area size is two double-words (128 bits).

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

- if boot in Flash is done in Bank 1, FLASH\_PCROP1SR and FLASH\_PCROP1ER registers must be programmed with:
  - PCROP1\_STRT = 0xC5F0.
  - PCROP1\_END = 0xE000.
- If the two banks are swapped, the protection must apply to bank 2, and FLASH\_PCROP2SR and FLASH\_PCROP2ER register must be programmed with:
  - PCROP2\_STRT = 0xC5F0.
  - PCROP2\_END = 0xE000.

Any read access performed through the D-bus to a PCROP protected area will trigger RDERR flag error.

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

For previous example, due to erase by page, all pages from page 0x62 to 0x70 are protected in case of page erase. (All addresses from 0x0806 2000 to 0x080 70FFF cannot be erased).

Deactivation of PCROP can only occurs 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 area, the options programming is launched but PCROP area stays unchanged. On the contrary, it is possible to increase the PCROP area.

When option bit PCROP\_RDP is cleared, when the RDP is changing from level 1 to level 0, Full Mass Erase is replaced by Partial Mass Erase in order to keep the PCROP area (refer to [Changing the Read protection level](#)). In this case, PCROP1/2\_STRT and PCROP1/2\_END are also not erased.

*Note: It is recommended to align PCROP area with page granularity when using PCROP\_RDP, or to leave free the rest of the page where PCROP zone starts or ends.*

**Table 14. PCROP protection<sup>(1)</sup>**

PCROPx registers values (x = 1,2)	PCROP protection area
PCROPx_offset_strt > PCROPx_offset_end	No PCROP area.
PCROPx_offset_strt < PCROPx_offset_end	The area between PCROPx_offset_strt and PCROPx_offset_end is protected. It is possible to write: <ul style="list-style-type: none"> <li>– PCROPx_offset_strt with a lower value</li> <li>– PCROPx_offset_end with a higher value.</li> </ul>

- When DBANK=1, the minimum PCROP area size is 2xdouble words: PCROPx\_offset\_strt and PCROPx\_offset\_end.  
When DBANK=0, the minimum PCROP area size is 2x(2xdouble words): PCROPx\_offset\_strt and PCROPx\_offset\_end.  
When DBANK=1, it is the user's responsibility to make sure no overlapping occurs on the PCROP zones.

### 3.5.3 Write protection (WRP)

The user area in Flash memory can be protected against unwanted write operations.

Depending on the DBANK option bit configuration, it allows either to specify:

- In Single-bank mode (DBANK=0): four write-protected (WRP) areas can be defined in each bank, with page size (8 KByte) granularity.
- In Dual-bank mode (DBANK=1): two write-protected (WRP) areas can be defined in each bank, with page (4 KByte) granularity.

Each area is defined by a start page offset and an end page offset related to the physical Flash bank base address. These offsets are defined in the WRP address registers: *Flash WRP1 area A address register (FLASH\_WRP1AR)*, *Flash WRP1 area B address register (FLASH\_WRP1BR)*, *Flash WRP2 area A address register (FLASH\_WRP2AR)*, *Flash WRP2 area B address register (FLASH\_WRP2BR)*.

#### Dual-bank mode (DBANK=1)

The bank "x" WRP "y" area (x=1,2 and y=A,B) is defined from the address: Bank "x" Base address + [WRPxy\_STRT x 0x1000] (included) to the address: Bank "x" Base address + [(WRPxy\_END+1) x 0x1000] (excluded).

#### Single-bank mode (DBANK=0)

The WRPx "y" area (x=1,2 and y=A,B) is defined from the address: Base address + [WRPy\_STRT x 0x2000] (included) to the address: Base address + [(WRPy\_END+1) x 0x2000] (excluded).

For example, to protect by WRP from the address 0x0806 2800 (included) to the address 0x0807 07FF (included):

- If boot in Flash is done in Bank 1, FLASH\_WRP1AR register must be programmed with:
  - WRP1A\_STRT = 0x62.
  - WRP1A\_END = 0x70.
 WRP1B\_STRT and WRP1B\_END in FLASH\_WRP1BR can be used instead (area "B" in Bank 1).
- If the two banks are swapped, the protection must apply to bank 2, and FLASH\_WRP2AR register must be programmed with:
  - WRP2A\_STRT = 0x62.
  - WRP2A\_END = 0x70.
  - WRP2A\_STRT = 0xC5.
  - WRP2A\_END = 0xE0.
 WRP2B\_STRT and WRP2B\_END in FLASH\_WRP2BR can be used instead (area "B" in Bank 2).

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 memory read protection level is selected (RDP level = 1), it is not possible to program or erase Flash memory if the CPU debug features are connected (JTAG or single wire) or boot code is being executed from RAM or System Flash, even if WRP is not activated.

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

*Note:* When DBANK=0, it is the user’s responsibility to make sure that no overlapping occurs on the WRP zone.

**Table 15. WRP protection**

WRP registers values (x=1/2 y= A/B)	WRP protection area
WRPxy_STRT = WRPxy_END	Page WRPxy is protected.
WRPxy_STRT > WRPxy_END	No WRP area.
WRPxy_STRT < WRPxy_END	- The pages from WRPxy_STRT to WRPxy_END are protected.

### 3.6 FLASH interrupts

**Table 16. Flash interrupt request**

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 error	RDERR	Write RDERR=1	RDERRIE
ECC correction	ECCC	Write ECCC=1	ECCIE

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



### 3.7 FLASH registers

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

Address offset: 0x00

Reset value: 0x0000 0600

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.	SLEEP_PD	RUN_PD	DCRST	ICRST	DCEN	ICEN	PRFTEN	Res.	Res.	Res.	Res.	LATENCY [3:0]			
	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

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

**Bit 14 SLEEP\_PD:** Flash Power-down mode during Sleep or Low-power sleep mode  
 This bit determines whether the Flash memory is in Power-down mode or idle mode when the device is in Sleep or Low-power sleep mode.  
 0: Flash in idle mode during Sleep and Low-power sleep modes  
 1: Flash in Power-down mode during Sleep and Low-power sleep modes

**Caution:** The Flash must not be put in power-down while a program or an erase operation is on-going.

**Bit 13 RUN\_PD:** Flash Power-down mode during Run or Low-power run mode  
 This bit is write-protected with FLASH\_PDKEYR.  
 This bit determines whether the Flash memory is in Power-down mode or idle mode when the device is in Run or Low-power run mode. The Flash memory can be put in Power-down mode only when the code is executed from RAM. The Flash must not be accessed when RUN\_PD is set.  
 0: Flash in idle mode  
 1: Flash in Power-down mode

**Caution:** The Flash must not be put in power-down while a program or an erase operation is on-going.

**Bit 12 DCRST:** Data cache reset  
 0: Data cache is not reset  
 1: Data cache is reset  
 This bit can be written only when the data cache is disabled.

**Bit 11 ICRST:** Instruction cache reset  
 0: Instruction cache is not reset  
 1: Instruction cache is reset  
 This bit can be written only when the instruction cache is disabled.

**Bit 10 DCEN:** Data cache enable  
 0: Data cache is disabled  
 1: Data cache is enabled

- Bit 9 **ICEN**: Instruction cache enable
  - 0: Instruction cache is disabled
  - 1: Instruction cache is enabled
- Bit 8 **PRFTEN**: Prefetch enable
  - 0: Prefetch disabled
  - 1: Prefetch enabled
- Bits 7:4 Reserved, must be kept at reset value.
- Bits 3:0 **LATENCY[0]**: Latency
  - These bits represent the ratio of the SYSCLK (system clock) period to the Flash access time.
  - 0000: Zero wait state
  - 0001: One wait state
  - 0010: Two wait states
  - 0011: Three wait states
  - 0100: Four wait states
  - ...1111: Fifteen wait states

### 3.7.2 Flash Power-down key register (FLASH\_PDKEYR)

Address offset: 0x04

Reset value: 0x0000 0000

Access: no wait state, word access

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

- Bits 31:0 **PDKEYR**: Power-down in Run mode Flash key
  - The following values must be written consecutively to unlock the RUN\_PD bit in FLASH\_ACR:
  - PDKEY1: 0x0415 2637
  - PDKEY2: 0xFAFB FCFD



### 3.7.3 Flash key register (FLASH\_KEYR)

Address offset: 0x08

Reset value: 0x0000 0000

Access: no wait state, word access

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

Bits 31:0 **KEYR**: Flash key

The following values must be written consecutively to unlock the FLASH\_CR register allowing Flash programming/erasing operations:

KEY1: 0x4567 0123

KEY2: 0xCDEF 89AB

### 3.7.4 Flash option key register (FLASH\_OPTKEYR)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: no wait state, word access

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

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

The following values must be written consecutively to unlock the FLASH\_OPTR register allowing option byte programming/erasing operations:

KEY1: 0x0819 2A3B

KEY2: 0x4C5D 6E7F

### 3.7.5 Flash status register (FLASH\_SR)

Address offset: 0x10

Reset value: 0x0000 0000

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

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

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **PEMPTY**: Program EMPTY

Set by hardware on power-on reset or after OBL\_LAUNCH command execution if the Flash is not programmed and the user intends to boot from the main Flash. Cleared by hardware on power-on reset or after OBL\_LAUNCH command execution if the Flash is programmed and the user intends to boot from main Flash. This bit can also be set and cleared by software.

1: The bit value is toggling

0: No effect

This bit can be set to clear the Program Empty bit if an OBL\_LAUNCH is done by software after Flash programming (boot in main Flash selected). It finally forces the boot in the main Flash, without loosing the debugger connection.

Bit 16 **BSY**: Busy

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

Bit 15 **OPTVERR**: Option validity error

Set by hardware when the options read may not be the one configured by the user. If option haven't been properly loaded, OPTVERR is set again after each system reset.

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 (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 **MISERR**: Fast programming data miss error  
In fast programming mode, 32 double words must be sent to Flash successively, and the new data must be sent to the Flash logic control before the current data is fully programmed. MISERR 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, MISERR or FASTERR is set due to a previous programming error.  
Set also when trying to perform bank erase when DBANK=0.  
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 64-bit Flash memory row in case of standard programming, or if there is a change of page during fast programming.  
Cleared by writing 1.
- Bit 4 **WRPERR**: Write protection error  
Set by hardware when an address to be erased/programmed belongs to a write-protected part (by WRP, PCROP or RDP level 1) of the Flash memory.  
Cleared by writing 1.
- Bit 3 **PROGERR**: Programming error  
Set by hardware when a double-word address to be programmed contains a value different from '0xFFFF FFFF' before programming, except if the data to write is '0x0000 0000'.  
Cleared by writing 1.
- Bit 2 Reserved, must be kept at reset value.
- Bit 1 **OPERR**: Operation error  
Set by hardware when a Flash memory operation (program / erase) completes unsuccessfully.  
This bit is set only if error interrupts are enabled (ERRIE = 1).  
Cleared by writing '1'.
- Bit 0 **EOP**: End of operation  
Set by hardware when one or more Flash memory operation (programming / erase) has been completed successfully.  
This bit is set only if the end of operation interrupts are enabled (EOPIE = 1).  
Cleared by writing 1.

### 3.7.6 Flash control register (FLASH\_CR)

Address offset: 0x14

Reset value: 0xC000 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
LOCK	OPT LOCK	Res.	Res.	OBL_LAUNCH	RD ERRIE	ERR IE	EOP IE	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
MER2	Res.	Res.	Res.	BKER	PNB[7:0]								MER1	PER	PG
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

**Bit 31 LOCK:** FLASH\_CR Lock  
 This bit is set only. When set, the FLASH\_CR register is locked. It is cleared by hardware after detecting the unlock sequence.  
 In case of an unsuccessful unlock operation, this bit remains set until the next system reset.

**Bit 30 OPTLOCK:** Options Lock  
 This bit is set only. When set, all bits concerning user option in FLASH\_CR register and so option page are locked. 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:** Force 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 **START**: Start  
This bit triggers an erase operation when set. If MER1, MER2 and PER bits are reset and the STRT bit is set, an unpredictable behavior may occur without generating any error flag. This condition should be forbidden.  
This bit is set only by software, and is cleared when the BSY bit is cleared in FLASH\_SR.
- Bit 15 **MER2**: Bank 2 Mass erase  
This bit triggers the bank 2 mass erase (all bank 2 user pages) when set.
- Bits 14:12 Reserved, must be kept at reset value.
- Bit 11 **BKER**: Bank erase  
DBANK=1  
0: Bank 1 is selected for page erase  
1: Bank 2 is selected for page erase  
DBANK=0  
Reserved, must be kept cleared
- Bits 10:3 **PNB[7:0]**: Page number selection  
These bits select the page to erase:  
00000000: page 0  
00000001: page 1  
...  
11111111: page 255
- Bit 2 **MER1**: Bank 1 Mass erase  
This bit triggers the bank 1 mass erase (all bank 1 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

### 3.7.7 Flash ECC register (FLASH\_ECCR)

Address offset: 0x18

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	ECCD2	ECCD2	Res.	Res.	Res.	ECCC IE	Res.	SYSF_ECC	BK_ECC	ADDR_ECC[20:16]				
rc_w1	rc_w1	rc_w1	rc_w1				rw		r	r	r	r	r	r	r
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

DBANK=0

Set by hardware when two ECC errors have been detected (only if ECCC/ECCC2/ECCD/ ECCD2 are previously cleared). When this bit is set, a NMI is generated.

Cleared by writing 1.

DBANK=1

Set by hardware when two ECC errors have been detected on 64-bits LSB (bits 63:0) (only if ECCC/ECCC2/ECCD/ ECCD2 are previously cleared). When this bit is set, a NMI is generated.

Cleared by writing 1.

Bit 30 **ECCC**: ECC correction

Set by hardware when one ECC error has been detected and corrected (only if ECCC/ECCC2/ECCD/ECCD2 are previously cleared). An interrupt is generated if ECCIE is set.

Cleared by writing 1.

Bit 29 **ECCD2**: ECC2 detection

DBANK=0

Set by hardware when two ECC errors have been detected on 64-bits MSB (bits127:64). This bit is set (only if ECCC/ECCC2/ECCD/ECCD2 are previously cleared). When this bit is set, a NMI is generated.

Cleared by writing 1.

DBANK=1

Reserved, must be kept at reset value.

Bit 28 **ECCC2**: ECC correction

DBANK=0

Set by hardware when one ECC error has been detected and corrected on 64-bits MSB (bits127:64). This bit is set (only if ECCC/ECCC2/ECCD/ECCD2 are previously cleared). An interrupt is generated if ECCIE is set.

Cleared by writing 1.

DBANK=1

Reserved, must be kept at reset value.

Bits 27:25 Reserved, must be kept at reset value.



- Bit 24 **ECCIE**: ECC correction interrupt enable  
0: ECC interrupt disabled  
1: ECC interrupt enabled.  
  
DBANK=0  
This bit enables the interrupt generation when the ECC or ECC2 bits in the FLASH\_ECCR register are set.  
DBANK=1  
This bit enables the interrupt generation when the ECC bit in the FLASH\_ECCR register is set.
- Bit 23 Reserved, must be kept at reset value.
- Bit 22 **SYSF\_ECC**: System Flash ECC fail  
This bit indicates that the ECC error correction or double ECC error detection is located in the System Flash.
- Bit 21 **BK\_ECC**: ECC fail bank  
DBANK=1  
This bit indicates which bank is concerned by the ECC error correction or by the double ECC error detection.  
0: bank 1  
1: bank 2  
DBANK=0  
If SYSF\_ECC is 1, it indicates which bank is concerned by the ECC error  
If SYSF\_ECC is 0, reserved, must be kept cleared.
- Bit 19 **BK\_ECC**: ECC fail bank  
This bit indicates which bank is concerned by the ECC error correction or by the double ECC error detection.  
0: bank 1  
1: bank 2
- Bits 20:0 **ADDR\_ECC**: ECC fail address  
DBANK=0  
This bit indicates which address in the Flash memory is concerned by the ECC error correction or by the double ECC error detection.  
DBANK=1  
This bit indicates which address in the bank is concerned by the ECC error correction or by the double ECC error detection.

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

Address offset: 0x20

Reset value: 0xFFEF F8AA. Register bits are loaded with values from Flash memory at OBL.

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.	n BOOT0	nSW BOOT0	SRAM2 _RST	SRAM2 _PE	nBOOT 1	DBANK	DB1M	BFB2	WWDG _SW	IWGD STDBY	IWDG _STOP	IWDG _SW
				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	Res.	BOR_LEV[2:0]			RDP[7:0]							
	rw	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 **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 Erase when system reset  
 0: SRAM2 erased when a system reset occurs  
 1: SRAM2 is 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, this bit selects boot mode from the Flash main memory, SRAM1 or the System memory. Refer to [Section 2.6: Boot configuration](#).

Bit 22 **DBANK**:  
 0: Single-bank mode with 128 bits data read width  
 1: Dual-bank mode with 64 bits data  
 This bit can only be written when PCROPA/B is disabled.

Bit 21 **DB1M**: Dual-bank on 1 Mbyte Flash memory devices  
 0: 1 Mbyte single Flash contiguous address in bank1  
 1: 1 Mbyte dual-bank Flash with contiguous addresses.  
 When DB1M is set, a hard Fault is generated when the requested address goes over 1 Mbyte.

Bit 20 **BFB2**: Dual-bank boot  
 0: Dual-bank boot disable  
 1: Dual-bank boot enable

- Bit 19 **WWDG\_SW**: Window watchdog selection  
0: Hardware window watchdog  
1: Software window watchdog
- Bit 18 **IWDG\_STDBY**: Independent watchdog counter freeze in Standby mode  
0: Independent watchdog counter is frozen in Standby mode  
1: Independent watchdog counter is running in Standby mode
- Bit 17 **IWDG\_STOP**: Independent watchdog counter freeze in Stop mode  
0: Independent watchdog counter is frozen in Stop mode  
1: Independent watchdog counter is running in Stop mode
- Bit 16 **IDWG\_SW**: Independent watchdog selection  
0: Hardware independent watchdog  
1: Software independent watchdog
- Bit 15 Reserved, must be kept cleared
- 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 generate 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
- Bit 11 Reserved, must be kept cleared
- Bits10:8 **BOR\_LEV**: BOR reset Level  
These bits contain the VDD supply level threshold that activates/releases the reset.  
000: BOR Level 0. Reset level threshold is around 1.7 V  
001: BOR Level 1. Reset level threshold is around 2.0 V  
010: BOR Level 2. Reset level threshold is around 2.2 V  
011: BOR Level 3. Reset level threshold is around 2.5 V  
100: BOR Level 4. Reset level threshold is around 2.8 V
- Bits 7:0 **RDP**: 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 [Section : Level 1: Read protection](#) for more details.*

### 3.7.9 Flash PCROP1 Start address register (FLASH\_PCROP1SR)

Address offset: 0x24

Reset value: 0xFFFF FFFF. Register bits are loaded with values from Flash memory at OBL.

Access: no wait state when no Flash memory operation is on going; 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.	PCROP1_STRT
															r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCROP1_STRT[16:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:17 Reserved, must be kept

Bits 16:0 **PCROP1\_STRT**: PCROP area start offset

DBANK=1

PCROP1\_STRT contains the first double-word of the PCROP area for bank1.

DBANK=0

PCROP1\_STRT contains the first 2xdouble-word of the PCROP area for all memory.

### 3.7.10 Flash PCROP1 End address register (FLASH\_PCROP1ER)

Address offset: 0x28

Reset value: 0xFFFF XXXX. Register bits are loaded with values from Flash memory at OBL.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCROP_RDP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1_END
r/s															r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCROP1_END[16:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bit 31 **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 30:17 Reserved, must be kept cleared

Bits 16:0 **PCROP1\_END**: Bank 1 PCROP area end offset  
 DBANK=1  
 PCROP1\_END contains the last double-word of the bank 1 PCROP area.  
 DBANK=0  
 PCROP1\_END contains the last 2x double-word PCROP area for all memory.

### 3.7.11 Flash WRP1 area A address register (FLASH\_WRP1AR)

Address offset: 0x2C

Reset value: 0xFFXX FFXX. Register bits are loaded with values from Flash memory at OBL.

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.	WRP1A_END[7:0]							
								r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1A_STRT[7:0]							
								r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:24 Reserved, must be kept cleared

Bits 23:16 **WRP1A\_END**: WRP first area “A” end offset  
 DBANK=1  
 WRP1A\_END contains the last page of WRP first area in bank1.  
 DBANK=0  
 WRP1A\_END contains the last page of WRP first area for all memory.

Bits 15:8 Reserved, must be kept cleared

Bits 7:0 **WRP1A\_STRT**: WRP first area “A” start offset  
 DBANK=1  
 WRP1A\_STRT contains the first page of WRP first area for bank1.  
 DBANK=0  
 WRP1A\_STRT contains the first page of WRP first area for all memory.

### 3.7.12 Flash WRP1 area B address register (FLASH\_WRP1BR)

Address offset: 0x30

Reset value: 0xFFXX FFXX. Register bits are loaded with values from Flash memory at OBL.

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.	WRP1B_END[7:0]							
								r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_STRT[7:0]							
								r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:24 Reserved, must be kept cleared

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

DBANK=1

WRP1B\_END contains the last page of the WRP second area for bank1.

DBANK=0

WRP1B\_END contains the last page of the WPR second area for all memory.

Bits 15:8 Reserved, must be kept cleared

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

DBANK=1

WRP1B\_STRT contains the last page of the WRP second area for bank1.

DBANK=0

WRP1B\_STRT contains the last page of the WPR second area for all memory.

### 3.7.13 Flash PCROP2 Start address register (FLASH\_PCROP2SR)

Address offset: 0x44

Reset value: 0xFFFF FFFF

Access: no wait state when no Flash memory operation is on going; 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.	PCRO P2_ST RT
															r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCROP2_STRT[16:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:17 Reserved, must be kept cleared

Bits 16:0 **PCROP2\_STRT**: PCROP area start offset

DBANK=1

PCROP2\_STRT contains the first double-word of the PCROP area for bank 2.

DBANK=0

PCROP2\_STRT contains the first double-word PCROP area for all memory.

### 3.7.14 Flash PCROP2 End address register (FLASH\_PCROP2ER)

Address offset: 0x48

Reset value: 0xXFFX XXXX

Access: no wait state when no Flash memory operation is on going; 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.	PCRO P2_EN D
															r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCROP2_END[16:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:17 Reserved, must be kept cleared

Bits 23:16 **PCROP2\_END**: PCROP area end offset

DBANK=1

PCROP2\_END contains the last double-word of the PCROP area for bank2.

DBANK=0

PCROP2\_END contains the last 2xdouble-word of the PCROP area for all the memory.

### 3.7.15 Flash WRP2 area A address register (FLASH\_WRP2AR)

Address offset: 0x4C

Reset value: 0xFFXX FFXX

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.	WRP2A_END[7:0]							
								r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2A_STRT[7:0]							
								r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:24 Reserved, must be kept cleared

Bits 23:16 **WRP2A\_END**: WRP first area “A” end offset  
 DBANK=1  
 WRP2A\_END contains the last page of the WRP first area for bank2.  
 DBANK=0  
 WRP2A\_END contains the last page of the WRP third area for all memory.

Bits 15:8 Reserved, must be kept cleared

Bits 7:0 **WRP2A\_STRT**: WRP first area “A” start offset  
 DBANK=1  
 WRP2A\_STRT contains the first page of the WRP first area for bank2.  
 DBANK=0  
 WRP2A\_STRT contains the first page of the WRP third area for all memory.

### 3.7.16 Flash WRP2 area B address register (FLASH\_WRP2BR)

Address offset: 0x50

Reset value: 0xFFXX FFX

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.	WRP2B_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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2B_STRT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept cleared

Bits 23:16 **WRP2B\_END**: WRP second area “B” end offset  
 DBANK=1  
 WRP2B\_END contains the last page of the WRP second area for bank2.  
 DBANK=0  
 WRP2B\_END contains the last page of the WRP fourth area for all memory.

Bits 15:8 Reserved, must be kept cleared

Bits 7:0 **WRP2B\_STRT**: WRP second area “B” start offset  
 DBANK=1  
 WRP2B\_STRT contains the first page of the WRP second area for bank2.  
 DBANK=0  
 WRP2B\_STRT contains the first page of the WRP second area for all memory.



### 3.7.17 Flash configuration register (FLASH\_CFGR)

Address offset: 0x130

Reset value: 0x0000 0000

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

Bits 31:1 Reserved, must be kept cleared

Bit 0 **LVEN**: Low voltage enable

This bit is set and cleared by software. This bit must be used only in case of external SMPS.

0: Flash low voltage disable

1: Flash low voltage enabled. Before setting this bit, it is recommended to:

- Switch the voltage scaling to range 2
- Ensure that the minimum VDD12 is 1.08 V
- When the external SMPS is ON, the LVEN bit can be set.

3.7.18 FLASH register map

Table 17. 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			
0x00	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEP_PD	RUN_PD	DCRST	ICRST	DCEN	ICEN	PRFTEN	Res.	Res.	Res.	Res.	LATENCY [3:0]						
	Reset value																		0	0	0	0	1	1	0					0	0	0	0			
0x04	FLASH_PDKEYR	PDKEYR[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		
0x08	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		
0x0C	FLASH_OPTKEYR	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		
0x10	FLASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY	BSY	OPTVERR	RDERR	Res.	Res.	Res.	Res.	Res.	FASTERR	MISERR	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP		
	Reset value															X	0	0	0						0	0	0	0	0	0	0	0	0	0		
0x14	FLASH_CR	LOCK	OPTLOCK	Res.	Res.	OBL_LAUNCH	RDERRIE	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	FSTPG	OPTSTRT	STRT	MER2	Res.	Res.	Res.	BKER	PNB[7:0]							MER1	PER	PG				
	Reset value	1	1			0	0	0	0						0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0		
0x18	FLASH_ECCR	ECCD	ECCC	ECCD2	ECCC2	Res.	Res.	Res.	ECCEIE	Res.	SYSF_ECC	BK_ECC	ADDR_ECC[20:0]																							
	Reset value	0	0	0	0				0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x20	FLASH_OPTR	Res.	Res.	Res.	Res.	nBOOT0	nSWBOOT0	SRAM2_RST	SRAM2_PE	nBOOT1	DBANK	DB1M	BFB2	WWDG_SW	IWDG_STBY	IWDG_STOP	IWDG_SW	Res.	nRST_SHDW	nRST_STDB	nRST_STOP	Res.	BOR_LEV[2:0]			RDP[7:0]										
	Reset value					X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
0x24	FLASH_PCROP1SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1_STRT[16:0]																	
	Reset value																		1	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
0x28	FLASH_PCROP1ER	PCROP_RDP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1_END[16:0]																	
	Reset value	X																	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
0x2C	FLASH_WRP1AR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value									X	X	X	X	X	X	X	X	X																		



Table 17. Flash interface - register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
0x30	FLASH_WRP1BR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_END[7:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_STRT[7:0]											
	Reset value									X	X	X	X	X	X	X	X										X	X	X	X	X	X	X	X					
0x44	FLASH_PCROP2SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP2_STRT[16:0]																						
	Reset value																1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X						
0x48	FLASH_PCROP2ER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP2_END[16:0]																						
	Reset value																0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X						
0x4C	FLASH_WRP2AR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2A_END[7:0]							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					
0x50	FLASH_WRP2BR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2B_END[7:0]							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					
0x130	FLASH_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LVEN				
	Reset value																																		0				

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.

## 4 Firewall (FW)

### 4.1 Introduction

The Firewall is made to protect a specific part of code or data into the Non-Volatile Memory, and/or to protect the Volatile data into the SRAM 1 from the rest of the code executed outside the protected area.

### 4.2 Firewall main features

- The code to protect by the Firewall (Code Segment) may be located in:
  - The Flash memory map
  - The SRAM 1 memory, if declared as an executable protected area during the Firewall configuration step.
- The data to protect can be located either
  - in the Flash memory (non-volatile data segment)
  - in the SRAM 1 memory (volatile data segment)

The software can access these protected areas once the Firewall is opened. The Firewall can be opened or closed using a mechanism based on “call gate” (Refer to [Opening the Firewall](#)).

The start address of each segment and its respective length must be configured before enabling the Firewall (Refer to [Section 4.3.5: Firewall initialization](#)).

Each illegal access into these protected segments (if the Firewall is enabled) generates a reset which immediately kills the detected intrusion.

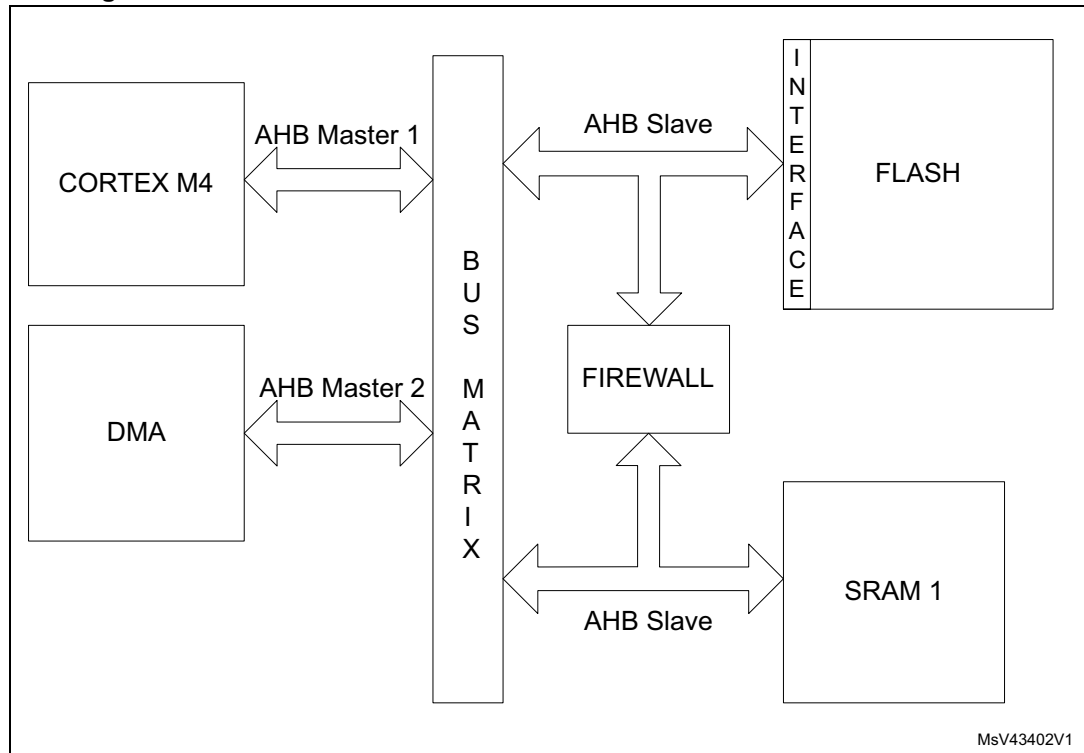
Any DMA access to protected segments is forbidden whatever the Firewall state (opened or closed). It is considered as an illegal access and generates a reset.

### 4.3 Firewall functional description

#### 4.3.1 Firewall AMBA bus snoop

The Firewall peripheral is snooping the AMBA buses on which the memories (volatile and non-volatile) are connected. A global architecture view is illustrated in [Figure 5](#).

Figure 5. STM32L4Rxxx and STM32L4Sxxx firewall connection schematics



#### 4.3.2 Functional requirements

There are several requirements to guaranty the highest security level by the application code/data which needs to be protected by the Firewall and to avoid unwanted Firewall alarm (reset generation).

##### Debug consideration

In debug mode, if the Firewall is opened, the accesses by the debugger to the protected segments are not blocked. For this reason, the Read out level 2 protection must be active in conjunction with the Firewall implementation.

If the debug is needed, it is possible to proceed in the following way:

- A dummy code having the same API as the protected code may be developed during the development phase of the final user code. This dummy code may send back coherent answers (in terms of function and potentially timing if needed), as the protected code should do in production phase.
- In the development phase, the protected code can be given to the customer-end under NDA agreement and its software can be developed in level 0 protection. The customer-

end code needs to embed an IAP located in a write protected segment in order to allow future code updates when the production parts will be Level 2 ROP.

### Write protection

In order to offer a maximum security level, the following points need to be respected:

- It is mandatory to keep a write protection on the part of the code enabling the Firewall. This activation code should be located outside the segments protected by the Firewall.
- The write protection is also mandatory on the code segment protected by the Firewall.
- The page including the reset vector must be write-protected.

### Interrupts management

The code protected by the Firewall must not be interruptible. It is up to the user code to disable any interrupt source before executing the code protected by the Firewall. If this constraint is not respected, if an interruption comes while the protected code is executed (Firewall opened), the Firewall will be closed as soon as the interrupt subroutine is executed. When the code returns back to the protected code area, a Firewall alarm will raise since the “call gate” sequence will not be applied and a reset will be generated.

Concerning the interrupt vectors and the first user page in the Flash memory:

- If the first user page (including the reset vector) is protected by the Firewall, the NVIC vector should be reprogrammed outside the protected segment.
- If the first user page is not protected by the Firewall, the interrupt vectors may be kept at this location.

There is no interrupt generated by the Firewall.

## 4.3.3 Firewall segments

The Firewall has been designed to protect three different segment areas:

### Code segment

This segment is located into the Flash memory. It should contain the code to execute which requires the Firewall protection. The segment must be reached using the “call gate” entry sequence to open the Firewall. A system reset is generated if the “call gate” entry sequence is not respected (refer to [Opening the Firewall](#)) and if the Firewall is enabled using the FWDIS bit in the system configuration register. The length of the segment and the segment base address must be configured before enabling the Firewall (refer to [Section 4.3.5: Firewall initialization](#)).

### Non-volatile data segment

This segment contains non-volatile data used by the protected code which must be protected by the Firewall. The access to this segment is defined into [Section 4.3.4: Segment accesses and properties](#). The Firewall must be opened before accessing the data in this area. The Non-Volatile data segment should be located into the Flash memory. The segment length and the base address of the segment must be configured before enabling the Firewall (refer to [Section 4.3.5: Firewall initialization](#)).

**Volatile data segment**

Volatile data used by the protected code located into the code segment must be defined into the SRAM 1 memory. The access to this segment is defined into the [Section 4.3.4: Segment accesses and properties](#). Depending on the Volatile data segment configuration, the Firewall must be opened or not before accessing this segment area. The segment length and the base address of the segment as well as the segment options must be configured before enabling the Firewall (refer to [Section 4.3.5: Firewall initialization](#)).

The Volatile data segment can also be defined as executable (for the code execution) or shared using two bit of the Firewall configuration register (bit VDS for the volatile data sharing option and bit VDE for the volatile data execution capability). For more details, refer to [Table 18](#).

**4.3.4 Segment accesses and properties**

All DMA accesses to the protected segments are forbidden, whatever the Firewall state, and generate a system reset.

**Segment access depending on the Firewall state**

Each of the three segments has specific properties which are presented in [Table 18](#).

**Table 18. Segment accesses according to the Firewall state**

Segment	Firewall opened access allowed	Firewall closed access allowed	Firewall disabled access allowed
Code segment	Read and execute	No access allowed. Any access to the segment (except the "call gate" entry) generates a system reset	All accesses are allowed (according to the Flash page protection properties in which the code is located)
Non-volatile data segment	Read and write	No access allowed	All accesses are allowed (according to the Flash page protection properties in which the code is located)
Volatile data segment	Read and Write Execute if VDE = 1 and VDS = 0 into the Firewall configuration register	No access allowed if VDS = 0 and VDE = 0 into the Firewall configuration register Read/write/execute accesses allowed if VDS = 1 (whatever VDE bit value) Execute if VDE = 1 and VDS = 0 but with a "call gate" entry to open the Firewall at first.	All accesses are allowed

The Volatile data segment is a bit different from the two others. The segment can be:

- Shared (VDS bit in the register)
 

It means that the area and the data located into this segment can be shared between the protected code and the user code executed in a non-protected area. The access is allowed whether the Firewall is opened or closed or disabled.

The VDS bit gets priority over the VDE bit, this last bit value being ignored in such a case. It means that the Volatile data segment can execute parts of code located there without any need to open the Firewall before executing the code.
- Execute
 

The VDE bit is considered as soon as the VDS bit = 0 in the FW\_CR register. If the VDS bit = 1, refer to the description above on the Volatile data segment sharing. If VDS = 0 and VDE = 1, the Volatile data segment is executable. To avoid a system reset generation from the Firewall, the “call gate” sequence should be applied on the Volatile data segment to open the Firewall as an entry point for the code execution.

### Segments properties

Each segment has a specific length register to define the segment size to be protected by the Firewall: CSL register for the Code segment length register, NVDSL for the Non-volatile data segment length register, and VDSL register for the Volatile data segment length register. Granularity and area ranges for each of the segments are presented in [Table 19](#).

**Table 19. Segment granularity and area ranges**

Segment	Granularity	Area range
Code segment	256 byte	2048 Kbytes - 256 bytes
Non-volatile data segment	256 byte	2048 Kbytes - 256 bytes
Volatile data segment	64 byte	192 Kbyte - 64 bytes

### 4.3.5 Firewall initialization

The initialization phase should take place at the beginning of the user code execution (refer to the [Write protection](#)).

The initialization phase consists of setting up the addresses and the lengths of each segment which needs to be protected by the Firewall. It must be done before enabling the Firewall, because the enabling bit can be written once. Thus, when the Firewall is enabled, it cannot be disabled anymore until the next system reset.

Once the Firewall is enabled, the accesses to the address and length segments are no longer possible. All write attempts are discarded.

A segment defined with a length equal to 0 is not considered as protected by the Firewall. As a consequence, there is no reset generation from the Firewall when an access to the base address of this segment is performed.

After a reset, the Firewall is disabled by default (FWDIS bit in the SYSCFG register is set). It has to be cleared to enable the Firewall feature.



Below is the initialization procedure to follow:

1. Configure the RCC to enable the clock to the Firewall module
2. Configure the RCC to enable the clock of the system configuration registers
3. Set the base address and length of each segment (CSSA, CSL, NVDSSA, NVDSL, VDSSA, VDSL registers)
4. Set the configuration register of the Firewall (FW\_CR register)
5. Enable the Firewall clearing the FWDIS bit in the system configuration register.

The Firewall configuration register (FW\_CR register) is the only one which can be managed in a dynamic way even if the Firewall is enabled:

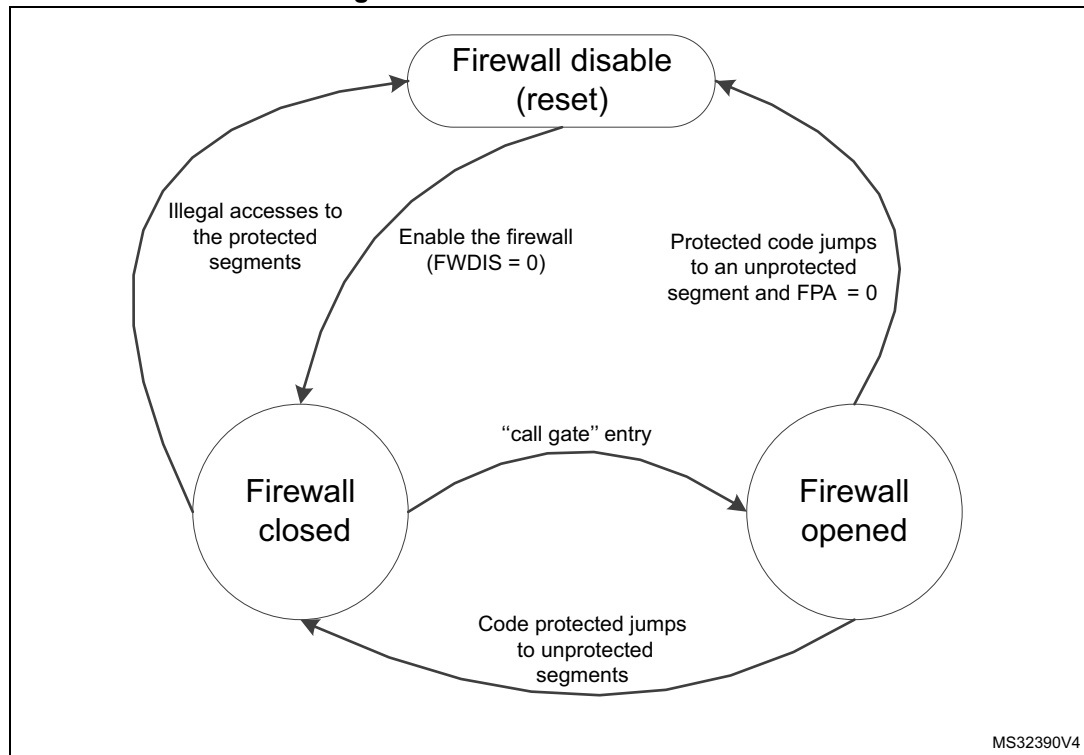
- when the Non-Volatile data segment is undefined (meaning the NVDSL register is equal to 0), the accesses to this register are possible whatever the Firewall state (opened or closed).
- when the Non-Volatile data segment is defined (meaning the NVDSL register is different from 0), the accesses to this register are only possible when the Firewall is opened.

### 4.3.6 Firewall states

The Firewall has three different states as shown in [Figure 6](#):

- Disabled: The FWDIS bit is set by default after the reset. The Firewall is not active.
- Closed: The Firewall protects the accesses to the three segments (Code, Non-volatile data, and Volatile data segments).
- Opened: The Firewall allows access to the protected segments as defined in [Section 4.3.4: Segment accesses and properties](#).

**Figure 6. Firewall functional states**



## Opening the Firewall

As soon as the Firewall is enabled, it is closed. It means that most of the accesses to the protected segments are forbidden (refer to [Section 4.3.4: Segment accesses and properties](#)). In order to open the Firewall to interact with the protected segments, it is mandatory to apply the “call gate” sequence described hereafter.

### “call gate” sequence

The “call gate” is composed of 3 words located on the first three 32-bit addresses of the base address of the code segment and of the Volatile data segment if it is declared as not shared (VDS = 0) and executable (VDE = 1).

- 1st word: Dummy 32-bit words always closed in order to protect the “call gate” opening from an access due to a prefetch buffer.
- 2nd and 3rd words: 2 specific 32-bit words called “call gate” and always opened.

To open the Firewall, the code currently executed must jump to the 2<sup>nd</sup> word of the “call gate” and execute the code from this point. The 2nd word and 3rd word execution must not be interrupted by any intermediate instruction fetch; otherwise, the Firewall is not considered open and comes back to a close state. Then, executing the 3<sup>rd</sup> word after receiving the intermediate instruction fetch would generate a system reset as a consequence.

As soon as the Firewall is opened, the protected segments can be accessed as described in [Section 4.3.4: Segment accesses and properties](#).

## Closing the Firewall

The Firewall is closed immediately after it is enabled (clearing the FWDIS bit in the system configuration register).

To close the Firewall, the protected code must:

- Write the correct value in the Firewall Pre Arm Flag into the FW\_CR register.
- Jump to any executable location outside the Firewall segments.

If the Firewall Pre Arm Flag is not set when the protected code jumps to a non protected segment, a reset is generated. This control bit is an additional protection to avoid an undesired attempt to close the Firewall with the private information not yet cleaned (see the note below).

*For security reasons, following the application for which the Firewall is used, it is advised to clean all private information from CPU registers and hardware cells.*

## 4.4 Firewall registers

### 4.4.1 Code segment start address (FW\_CSSA)

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.	ADD[23:16]							
								rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD[15:8]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

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

Bits 23:8 **ADD[23:8]**: code segment start address

The LSB bits of the start address (bit 7:0) are reserved and forced to 0 in order to allow a 256-byte granularity.

*Note: These bits can be written only before enabling the Firewall. Refer to [Section 4.3.5: Firewall initialization](#).*

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

### 4.4.2 Code segment length (FW\_CSL)

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.	LENG[21:16]					
										rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LENG[15:8]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	
rw															

Bits 31:22 Reserved, must be kept at the reset value.

Bits 21:8 **LENG[21:8]**: code segment length

LENG[21:8] selects the size of the code segment expressed in bytes but is a multiple of 256 bytes.

The segment area is defined from {ADD[23:8],0x00} to {ADD[23:8]+LENG[21:8], 0x00} - 0x01

*Note: If LENG[21:8] = 0 after enabling the Firewall, this segment is not defined, thus not protected by the Firewall.*

*These bits can only be written before enabling the Firewall. Refer to [Section 4.3.5: Firewall initialization](#).*

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

### 4.4.3 Non-volatile data segment start address (FW\_NVDSOA)

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.	ADD[23:16]							
								rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD[15:8]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

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

Bits 23:8 **ADD[23:8]**: Non-volatile data segment start address

The LSB bits of the start address (bit 7:0) are reserved and forced to 0 in order to allow a 256-byte granularity.

*Note: These bits can only be written before enabling the Firewall. Refer to [Section 4.3.5: Firewall initialization](#).*

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

### 4.4.4 Non-volatile data segment length (FW\_NVDSL)

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.	LENG[21:16]					
										rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LENG[15:8]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	
rw															

Bits 31:22 Reserved, must be kept at the reset value.

Bits 21:8 **LENG[21:8]**: Non-volatile data segment length

LENG[21:8] selects the size of the Non-volatile data segment expressed in bytes but is a multiple of 256 bytes.

The segment area is defined from {ADD[23:8],0x00} to {ADD[23:8]+LENG[21:8], 0x00} - 0x01

*Note: If LENG[21:8] = 0 after enabling the Firewall, this segment is not defined, thus not protected by the Firewall.*

*These bits can only be written before enabling the Firewall. Refer to [Section 4.3.5: Firewall initialization](#).*

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

### 4.4.5 Volatile data segment start address (FW\_VDSSA)

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.	ADD [17:16]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD[15:6]										Res.	Res.	Res.	Res.	Res.	Res.
rw															

Bits 31:18 Reserved, must be kept at the reset value.

Bit 17 **ADD[17]**: Volatile data segment start address

Bits 16:6 **ADD[16:6]**: Volatile data segment start address

The LSB bits of the start address (bit 5:0) are reserved and forced to 0 in order to allow a 64-byte granularity.

*Note: These bits can only be written before enabling the Firewall. Refer to [Section 4.3.5: Firewall initialization](#)*

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

### 4.4.6 Volatile data segment length (FW\_VDSL)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bits 31:18 Reserved, must be kept at the reset value.

Bit 17 **LENG[17]**: volatile data segment length

Bits 16:6 **LENG[16:6]**: volatile data segment length

LENG[16:6] selects the size of the volatile data segment expressed in bytes but is a multiple of 64 bytes.

The segment area is defined from {ADD[16:6],0x00} to {ADD[16:6]+LENG[16:6], 0x00} - 0x01

*Note: If LENG[17:6] = 0 after enabling the Firewall, this segment is not defined, thus not protected by the Firewall.*

*These bits can only be written before enabling the Firewall. Refer to [Section 4.3.5: Firewall initialization](#).*

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

### 4.4.7 Configuration register (FW\_CR)

Address offset: 0x20

Reset value: 0x0000 0000

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

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

Bit 2 **VDE**: Volatile data execution

0: Volatile data segment cannot be executed if VDS = 0

1: Volatile data segment is declared executable whatever VDS bit value

When VDS = 1, this bit has no meaning. The Volatile data segment can be executed whatever the VDE bit value.

If VDS = 1, the code can be executed whatever the Firewall state (opened or closed)

If VDS = 0, the code can only be executed if the Firewall is opened or applying the “call gate” entry sequence if the Firewall is closed.

Refer to [Segment access depending on the Firewall state](#).

Bit 1 **VDS**: Volatile data shared

0: Volatile data segment is not shared and cannot be hit by a non protected executable code when the Firewall is closed. If it is accessed in such a condition, a system reset will be generated by the Firewall.

1: Volatile data segment is shared with non protected application code. It can be accessed whatever the Firewall state (opened or closed).

Refer to [Segment access depending on the Firewall state](#).

Bit 0 **FPA**: Firewall prearm

0: any code executed outside the protected segment when the Firewall is opened will generate a system reset.

1: any code executed outside the protected segment will close the Firewall.

Refer to [Closing the Firewall](#).

This register is protected in the same way as the Non-volatile data segment (refer to [Section 4.3.5: Firewall initialization](#)).

### 4.4.8 Firewall register map

The table below provides the Firewall register map and reset values.

**Table 20. Firewall register map and reset values**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																	
0x0	FW_CSSA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADD																		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															
0x4	FW_CSL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LENG																		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															
0x8	FW_NVDSA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADD																		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														
0xC	FW_NVDSL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LENG																		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														
0x10	FW_VDSSA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADD																		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															
0x14	FW_VDSL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LENG																		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															
0x18		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	FW_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																
	Reset Value																																		0	0	0													

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.





## 5 Power control (PWR)

### 5.1 Power supplies

The STM32L4Rxxx and STM32L4Sxxx devices require a 1.71 V to 3.6 V operating supply voltage ( $V_{DD}$ ). Several peripherals are supplied through independent power domains:  $V_{DDA}$ ,  $V_{DDIO2}$ ,  $V_{DDUSB}$ ,  $V_{DDDSI}$ . Those supplies must not be provided without a valid operating supply on the  $V_{DD}$  pin.

- $V_{DD} = 1.71 \text{ V to } 3.6 \text{ V}$   
 $V_{DD}$  is the external power supply for the I/Os, the internal regulator and the system analog such as reset, power management and internal clocks. It is provided externally through VDD pins.
- $V_{DDA} = 1.62 \text{ V (ADCs/COMP)} / 1.8 \text{ V (DACs/OPAMP)} / 2.4 \text{ V (VREFBUF)} \text{ to } 3.6 \text{ V}$   
 $V_{DDA}$  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.  $V_{DDA}$  should be preferably connected to  $V_{DD}$  when these peripherals are not used.
- $V_{DD12} = 1.05 \text{ to } 1.32 \text{ V}$   
 $V_{DD12}$  is the external power supply bypassing the internal regulator when connected to an external SMPS. It is provided externally through  $V_{DD12}$  pins and only available on packages with the external SMPS supply option.  $V_{DD12}$  does not require any external decoupling capacitance and cannot support any external load.
- $V_{DDUSB} = 3.0 \text{ V to } 3.6 \text{ V}$   
 $V_{DDUSB}$  is the external independent power supply for USB transceivers. The  $V_{DDUSB}$  voltage level is independent from the  $V_{DD}$  voltage.  $V_{DDUSB}$  should be preferably connected to  $V_{DD}$  when the USB is not used.  
 The  $V_{DDUSB}$  power supply may not be present as a dedicated pin, but to be internally bonded to  $V_{DD}$ . For such devices,  $V_{DD}$  has to respect the  $V_{DDUSB}$  supply range when the USB is used.
- $V_{DDIO2} = 1.08 \text{ V to } 3.6 \text{ V}$   
 $V_{DDIO2}$  is the external power supply for 14 I/Os (Port G[15:2]). The  $V_{DDIO2}$  voltage level is independent from the  $V_{DD}$  voltage and should preferably be connected to  $V_{DD}$  when PG[15:2] are not used.
- $V_{DDDSI}$  is an independent DSI power supply dedicated to the DSI regulator and the MIPI DPHY. This supply must be connected to the global  $V_{DD}$ .
- $V_{CAPDSI}$  pin is the output of the DSI regulator (1.2 V) which must be connected externally to  $V_{DD12DSI}$ .
- $V_{DD12DSI}$  pin is used to supply the MIPI D-PHY, and to supply the clock and data lanes pins. An external capacitor of 2.2 uF must be connected on  $V_{DD12DSI}$  pin.
- $V_{BAT} = 1.55 \text{ V to } 3.6 \text{ V}$   
 $V_{BAT}$  is the power supply for RTC, external clock 32 kHz oscillator and backup registers (through power switch) when  $V_{DD}$  is not present.  $V_{BAT}$  is internally bonded to VDD for small packages without dedicated pin.
- $V_{REF-}, V_{REF+}$   
 $V_{REF+}$  is the input reference voltage for ADCs and DACs. It is also the output of the internal voltage reference buffer when enabled.

When  $V_{DDA} < 2\text{ V}$ ,  $V_{REF+}$  must be equal to  $V_{DDA}$ .

When  $V_{DDA} \geq 2\text{ V}$ ,  $V_{REF+}$  must be between  $2\text{ V}$  and  $V_{DDA}$ .

$V_{REF+}$  can be grounded when ADC and DAC are not active.

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

- $V_{REF+}$  around  $2.048\text{ V}$ . This requires  $V_{DDA}$  equal to or higher than  $2.4\text{ V}$ .
- $V_{REF+}$  around  $2.5\text{ V}$ . This requires  $V_{DDA}$  equal to or higher than  $2.8\text{ V}$ .

$V_{REF-}$  and  $V_{REF+}$  pins are not available on all packages. When not available on the package, they are bonded to  $V_{SSA}$  and  $V_{DDA}$ , respectively.

When the  $V_{REF+}$  is double-bonded with  $V_{DDA}$  in a package, the internal voltage reference buffer is not available and must be kept disable (refer to related device datasheet for packages pinout description).

$V_{REF-}$  must always be equal to  $V_{SSA}$ .

An embedded linear voltage regulator is used to supply the internal digital power  $V_{CORE}$ .  $V_{CORE}$  is the power supply for digital peripherals and memories.

Figure 7. STM32L4S5xx/R5xx and STM32L4S7xx/L4R7xx power supply overview

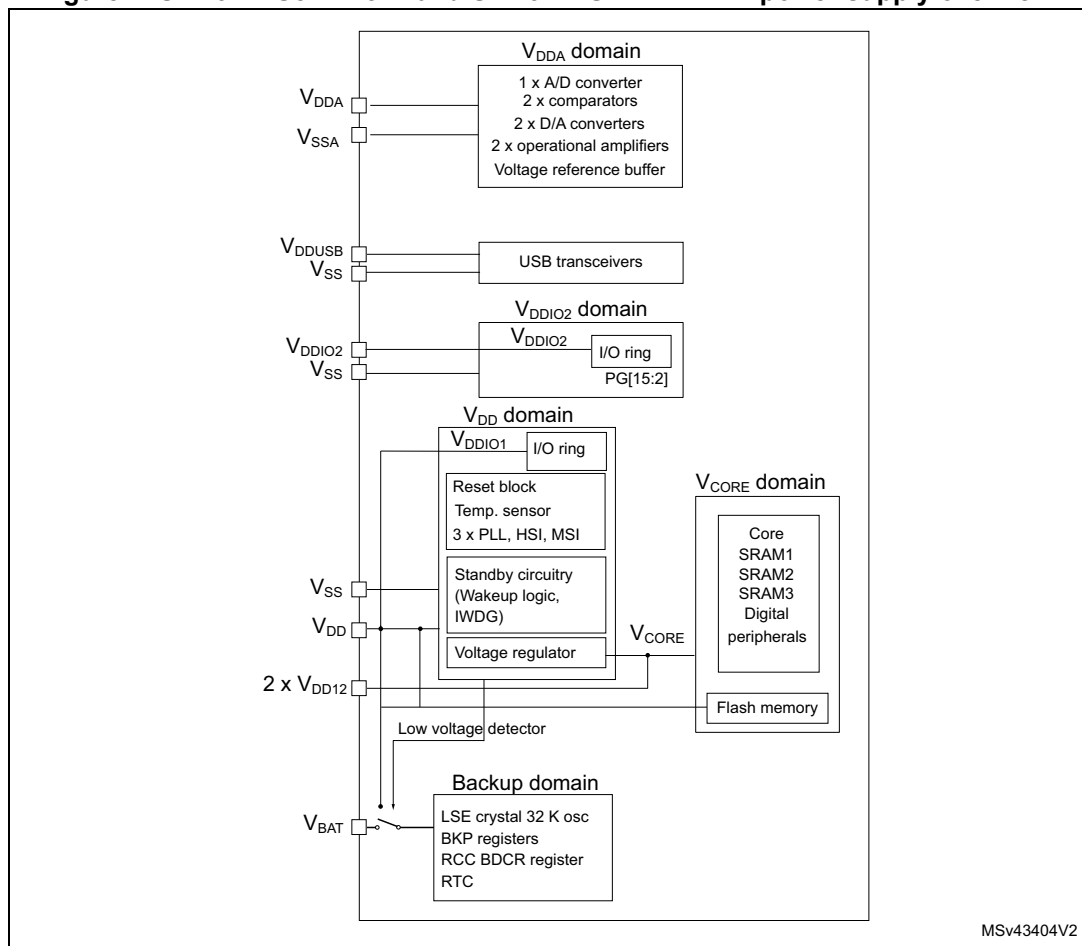
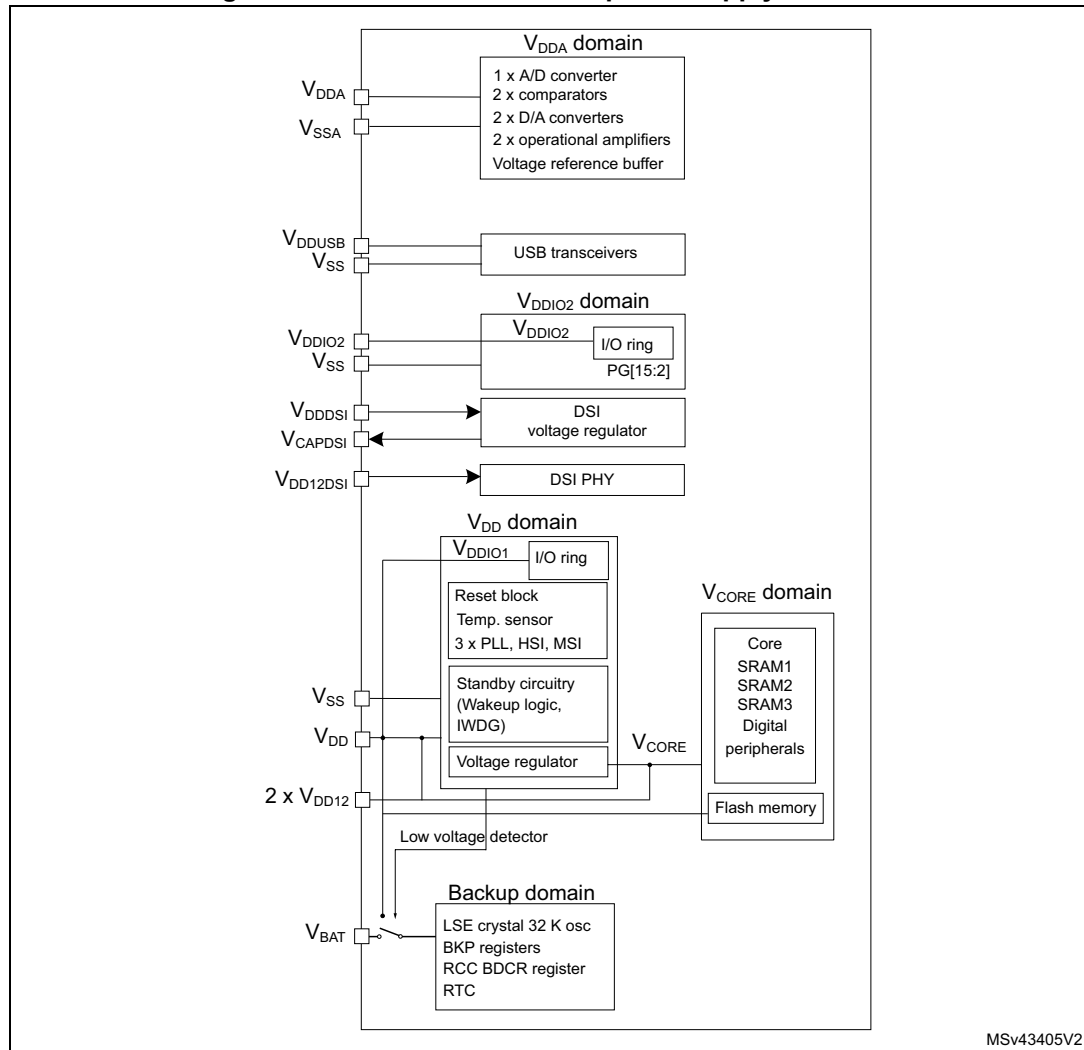


Figure 8. STM32L4S9xx/L4R9xx power supply overview



### 5.1.1 Independent analog peripherals supply

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

- The analog peripherals voltage supply input is available on a separate V<sub>DDA</sub> pin.
- An isolated supply ground connection is provided on V<sub>SSA</sub> pin.

The V<sub>DDA</sub> supply voltage can be different from V<sub>DD</sub>. The presence of V<sub>DDA</sub> must be checked before enabling any of the analog peripherals supplied by V<sub>DDA</sub> (A/D converter, D/A converter, comparators, operational amplifiers, voltage reference buffer).

The V<sub>DDA</sub> supply can be monitored by the Peripheral Voltage Monitoring, and compared with two thresholds (1.65 V for PVM3 or 2.2 V for PVM4), refer to [Section 5.2.3: Peripheral Voltage Monitoring \(PVM\)](#) for more details.

When a single supply is used, V<sub>DDA</sub> can be externally connected to V<sub>DD</sub> through the external filtering circuit in order to ensure a noise-free V<sub>DDA</sub> reference voltage.

### ADC and DAC reference voltage

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

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

The internal voltage reference is enabled by setting the ENVR bit in the [Section 24.3.1: 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 24: Voltage reference buffer \(VREFBUF\)](#) for further information.

### 5.1.2 Independent I/O supply rail

Some I/Os from Port G (PG[15:2]) are supplied from a separate supply rail. The power supply for this rail can range from 1.08 V to 3.6 V and is provided externally through the  $V_{DDIO2}$  pin. The  $V_{DDIO2}$  voltage level is completely independent from  $V_{DD}$  or  $V_{DDA}$ . The  $V_{DDIO2}$  pin is available only for some packages. Refer to the pinout diagrams or tables in the related device datasheet(s) for I/O list(s).

After reset, the I/Os supplied by  $V_{DDIO2}$  are logically and electrically isolated and therefore are not available. The isolation must be removed before using any I/O from PG[15:2], by setting the IOSV bit in the PWR\_CR2 register, once the  $V_{DDIO2}$  supply is present.

The  $V_{DDIO2}$  supply is monitored by the Peripheral Voltage Monitoring (PVM2) and compared with the internal reference voltage ( $3/4 V_{REFINT}$ , around 0.9V), refer to [Section 5.2.3: Peripheral Voltage Monitoring \(PVM\)](#) for more details.

### 5.1.3 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 OTG peripheral, by setting the USV bit in the PWR\_CR2 register, 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 ( $V_{REFINT}$ , around 1.2 V), refer to [Section 5.2.3: Peripheral Voltage Monitoring \(PVM\)](#) for more details.

### 5.1.4 Independent DSI supply

The DSI (Display Serial Interface) sub-system uses several power supply pins which are independent from the other supply pins:

- VDDDSI is an independent DSI power supply dedicated for DSI Regulator and MIPI D-PHY. This supply must be connected to global VDD.
- VCAPDSI pin is the output of DSI Regulator (1.2V) which must be connected externally to VDD12DSI.
- VDD12DSI pin is used to supply the MIPI D-PHY, and to supply clock and data lanes

pins. An external capacitor of 2.2 uF must be connected on VDD12DSI pin.

- VSSDSI pin is an isolated supply ground used for DSI sub-system.

If DSI functionality is not used at all, then:

- VDDDSI pin must be connected to global VDD.
- VCAPDSI pin must be connected externally to VDD12DSI but the external capacitor is no more needed. If the VDD12DSI is not available on a package, the VCAPDSI to be kept floating.
- VSSDSI pin must be grounded.

*Note:* VDDDSI and VDD12DSI pins are not available on all packages. When not available, they are bonded to VDD and VCAPDSI, respectively.

### 5.1.5 Battery backup domain

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

The VBAT pin powers the RTC 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 the value of  $t_{RSTTEMPO}$ ) and  $V_{DD} > V_{BAT} + 0.6$  V, a current may be injected into  $V_{BAT}$  through an internal diode connected between  $V_{DD}$  and the power switch ( $V_{BAT}$ ). If the power supply/battery connected to the VBAT pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the VBAT 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, PC14 and PC15, which can be used as GPIO pins
- PC13, PC14 and PC15, which can be configured by RTC or LSE (refer to [Section 42.3: RTC functional description on page 1403](#))
- PA0/RTC\_TAMP2 and PE6/RTC\_TAMP3 when they are configured by the RTC as tamper pins

*Note:* Due to the fact that 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 has to be limited to 2 MHz with a maximum load of 30 pF and these I/Os must not be used as a current source (e.g. to drive a LED).

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, PC14 and PC15 can be controlled only by RTC or LSE (refer to [Section 42.3: RTC functional description](#))
- PA0/RTC\_TAMP2 and PE6/RTC\_TAMP3 when they are 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. Enable the power interface clock by setting the PWREN bits in the [Section 6.4.19: APB1 peripheral clock enable register 1 \(RCC\\_APB1ENR1\)](#)
2. Set the DBP bit in the [Power control register 1 \(PWR\\_CR1\)](#) to enable access to the backup domain
3. Select the RTC clock source in the [Backup domain control register \(RCC\\_BDCR\)](#).
4. Enable the RTC clock by setting the RTCEN [15] bit in the [Backup domain control register \(RCC\\_BDCR\)](#).

### VBAT battery charging

When VDD 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 kOhm resistor or through a 1.5 kOhm resistor depending on the VBRS bit value in the PWR\_CR4 register.

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

## 5.1.6 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) in order to optimize the consumption depending on the system's maximum operating frequency (refer to [Section 6.2.9: Clock source frequency versus voltage scaling](#) and to [Section 3.3.3: 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 Stop 0 modes, both regulators are enabled and the main regulator (MR) supplies full power to the  $V_{CORE}$  domain (core, memories and digital peripherals).
- In Low-power run and Low-power sleep modes, the main regulator is off and the low-power regulator (LPR) supplies low-power to the  $V_{CORE}$  domain, preserving the contents of the registers, SRAM1, SRAM2 and SRAM3.
- In Stop 1 and Stop 2 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, SRAM1, SRAM2 and SRAM3.
- In Standby mode with SRAM2 content preserved (RRS bit is set in the PWR\_CR3

register), the main regulator (MR) is off and the low-power regulator (LPR) provides the supply to SRAM2 only. The core, digital peripherals (except Standby circuitry and backup domain) and SRAM1 are powered off.

- In Standby mode, both regulators are powered off. The contents of the registers, SRAM1, SRAM2 and SRAM3 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, SRAM1, SRAM2 and SRAM3 is lost, except for the backup domain.

### 5.1.7 VDD12 domain

VDD12 is intended to be connected with external SMPS (switched-mode power supply) to generate the  $V_{CORE}$  logic supply in Run, Sleep and Stop 0 modes only.

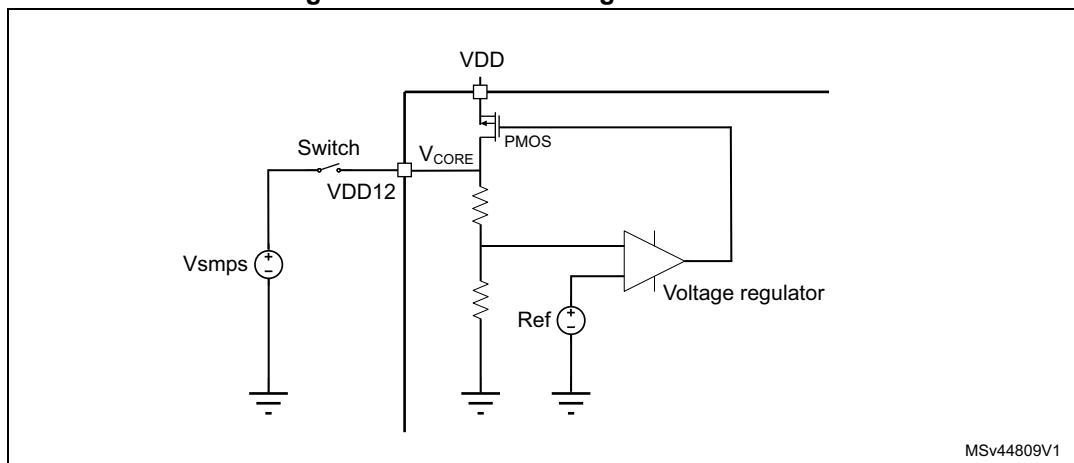
VDD12 pins correspond to the internal  $V_{CORE}$  powering the digital part of Core, RAMs, FLASH and peripherals. This significantly improves the power consumption with a gain from 50% or more depending of the SMPS performances.

The main benefit occurs in Run and Sleep modes whereas in Stop 0 mode, the gain is less significant.

The figure below shows a schematic to understand how the internal regulator stops supplying  $V_{CORE}$  when an external voltage VDD12 is provided.

As VDD12 shares the same pin as output of the internal regulator, applying a slightly higher voltage (typically +50 mV) on the VDD12 blocks, the PMOS and the regulator consumption is negligible.

**Figure 9. Internal main regulator overview**



A switch, controlled by the chosen GPIO, is inserted between the SMPS output and VDD12.

There are two possible states:

- Connected: Switch is closed so SMPS powers VDD12
- Disconnected: Switch is open and VDD12 is disconnected from SMPS output

Proper software management through GPIOs to enable/disable SMPS and to connect/disconnect SMPS through the switch, is required to conform with the rules described below. See also [Section 5.1.8: Dynamic voltage scaling management](#).



It is mandatory to respect the following rules to avoid any damage or instability on either digital parts or internal regulators:

- In Run, Sleep and Stop 0 modes, VDD12 can be connected and should respect
  - $VDD12 < 1.32\text{ V}$
  - $VDD12 \geq V_{CORE} + 50\text{ mV}$  giving for main regulator
    - Range 1 boost mode,  $V_{CORE} = 1.28\text{ V}$  so VDD12 should be greater than 1.33 V, but this cannot match previous rule  $VDD12 < 1.32\text{ V}$ , so it is not a functional use case with an external SMPS.
    - Range 1 normal mode,  $V_{CORE} = 1.2\text{ V}$  so VDD12 should be greater than 1.25 V.
    - Range 2,  $V_{CORE} = 1.0\text{ V}$  so VDD12 should be greater than 1.05 V
  - $VDD12 \geq 1.08\text{ V}$  in Range 2 when  $80\text{ MHz} \geq \text{SYSCLK frequency} \geq 26\text{ MHz}$
  - $VDD12 \geq 1.14\text{ V}$  in Range 2 when  $\text{SYSCLK frequency} > 80\text{ MHz}$
- In all other modes, such as LPRun, LPSleep, Stop 1, Stop 2, Standby and Shutdown modes, VDD12 must be disconnected from SMPS output. This means that the pin must be connected to an high impedance output:
  - VDD12 connected to HiZ (voltage is provided by internal regulators)
- Transitions of VDD12 from connected to disconnected is only allowed when SYSCLK frequency  $\leq 26\text{ MHz}$  to avoid to big voltage drop on main regulator side.

*Note:* In case of asynchronous reset while having the  $VDD12 \leq 1.25\text{ V}$ , VDD12 should switch to HiZ in less than regulator switching time from Range 2 to Range 1 (~1 us).

*Note:* For more details on VDD12 management, refer to AN4978 “Design recommendations for STM32L4xxx with external SMPS, for ultra-low-power applications with high performance”.

### 5.1.8 Dynamic voltage scaling management

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

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

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

- Range 1: High-performance range.

In Range1, the main regulator operates in two modes following the R1MODE bit in the PWR\_CR5 register:

- Main regulator Range 1 normal mode: provides a typical output voltage at 1.2 V. It is used when the system clock frequency is up to 80 MHz. The Flash access time for read access is minimum, write and erase operations are possible.
- Main regulator Range 1 boost mode: provides a typical output voltage at 1.28 V. It is used when the system clock frequency is up to 120 MHz. The Flash access time for read access is minimum, write and erase operations are possible. To optimize the power consumption it is recommended to select the range1 boost mode when the system clock frequency is greater than 80 MHz. See [Table 21](#).



Table 21. Range 1 boost mode configuration

System frequency	26 MHz < SYSCLK ≤ 80 MHz	80 MHz < SYSCLK ≤ 120 MHz
R1MODE bit configuration	1	0

- 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 26 MHz. The Flash access time for a read access is increased as compared to Range 1; write and erase operations are not possible.

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

The sequence to go from Range 1 (Normal/Boost) to Range 2 is:

1. In case of switching from Range 1 boost mode to Range 2, the system clock must be divided by 2 using the AHB prescaler before switching to a lower system frequency for at least 1us and then reconfigure the AHB prescaler.:
2. Reduce the system frequency to a value lower than 26 MHz
3. Adjust number of wait states according new frequency target in Range 2 (LATENCY bits in the FLASH\_ACR).
4. Program the VOS bits to "10" in the PWR\_CR1 register.

The sequence to go from Range 2 to Range 1 (normal/boost mode) is:

1. Program the VOS bits to "01" in the PWR\_CR1 register.
2. Wait until the VOSF flag is cleared in the PWR\_SR2 register.
3. Adjust number of wait states according new frequency target in Range 1 (LATENCY bits in the FLASH\_ACR).
4. Increase the system frequency by following below procedure:
  - If the system frequency is 26 MHz < SYSCLK ≤ 80 MHz:
    - Select the Range 1 normal mode by setting R1MODE bit in the PWR\_CR5 register.
    - Configure and switch to PLL for a new system frequency.
  - If the system frequency is SYSCLK > 80 MHz:
    - The system clock must be divided by 2 using the AHB prescaler before switching to a higher system frequency.
    - Select the Range 1 boost mode by clearing the R1MODE bit is in the PWR\_CR5 register,
    - Configure and switch to PLL for a new system frequency.
    - Wait for at least 1us and then reconfigure the AHB prescaler to get the needed HCLK clock frequency.

The sequence to switch from Range1 normal mode to Range1 boost mode is:

1. The system clock must be divided by 2 using the AHB prescaler before switching to a higher system frequency.
2. Clear the R1MODE bit is in the PWR\_CR5 register,
3. Adjust the number of wait states according to the new frequency target in range1 boost mode
4. Configure and switch to new system frequency.
5. Wait for at least 1 1us and then reconfigure the AHB prescaler to get the needed HCLK clock frequency.

The sequence to switch from Range1 boost mode to Range1 normal mode is:

1. Set the R1MODE bit is in the PWR\_CR5 register.
2. Adjust the number of wait states according new frequency target in Range1 default mode
3. Configure and switch to new system frequency.

When supplying VDD12 with an external SMPS, three states can be configured:

- *SMPS Range 1*: main regulator is in normal Range 1,  $V_{CORE}$  is supplied by external SMPS and higher than 1.25 V. SYSCLK frequency can be up to 120 MHz.
- *SMPS Range 2 high*: main regulator is in Range 2,  $V_{CORE}$  is supplied by external SMPS and higher than
  - 1.08 V when SYSCLK frequency  $\leq$  80 MHz
  - 1.14 V when SYSCLK frequency  $>$  80MHz
- *SMPS Range 2 low*: main regulator is in Range 2,  $V_{CORE}$  is supplied by external SMPS and higher than 1.05 V. Max SYSCLK frequency is 26 MHz without Flash write/erase operations.

In order to match the upper rules described in [Section 5.1.7: VDD12 domain](#), the transition sequences can only be one of the following:

- Range 1 to *SMPS Range 1*:
  1. Start SMPS converter (if not always enabled by HW).
  2. Check that SMPS converter output is at the correct level, like  $1.25\text{ V} \leq VDD12 < 1.32\text{ V}$ .
  3. Connect VDD12 to external SMPS converter through the switch.
  
- Range 2 to *SMPS Range 2 low and high*:
  1. Start SMPS (if not always enabled by HW).
  2. Check that SMPS output is at the correct level like  $1.05\text{ V} \leq VDD12 < 1.32\text{ V}$ .
  3. Connect VDD12 to external SMPS converter through the switch.
    - If  $1.08\text{ V} \leq VDD12$  (like *SMPS Range 2 high*), then the following steps can be applied:
      1. Set register FLASH\_CFGR bit LVEN to 1.
      2. Adjust the number of wait states in the FLASH\_ACR (up to max frequency of Range 1 refer to [Section 3.3.3: Read access latency](#)).
      3. Increase the system frequency up to the maximum allowed value for voltage Range 1 (like 80 MHz when  $1.08\text{ V} < VDD12 < 1.14\text{ V}$  or 120 MHz when  $VDD12 > 1.14\text{ V}$ ).
  
- *SMPS Range 1* to Range 1 or *SMPS Range 2 low and high* to Range 2:

1. If in Range 1, reduce the system frequency to a value lower or equal to 26 MHz.
2. Adjust number of wait states according new frequency target corresponding to voltage range (LATENCY bits in the FLASH\_ACR).
3. Set register FLASH\_CFGR bit LVEN to 0.
4. Disconnect VDD12 by opening the switch.
5. Stop SMPS (if required and not kept always enabled).

## 5.2 Power supply supervisor

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

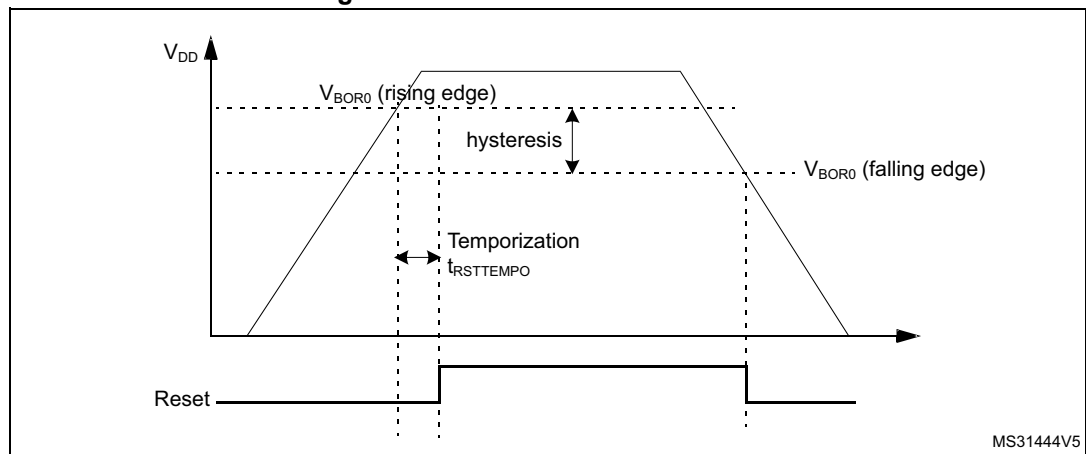
The device has an integrated power-on reset (POR) / power-down reset (PDR), coupled with a brown-out reset (BOR) circuitry. The BOR is active in all power modes except Shutdown mode, and cannot be disabled.

Five BOR thresholds can be selected through option bytes.

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.

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

Figure 10. Brown-out reset waveform



1. The reset temporization  $t_{RSTTEMPO}$  is present only for the BOR lowest threshold ( $V_{BOR0}$ ).

### 5.2.2 Programmable voltage detector (PVD)

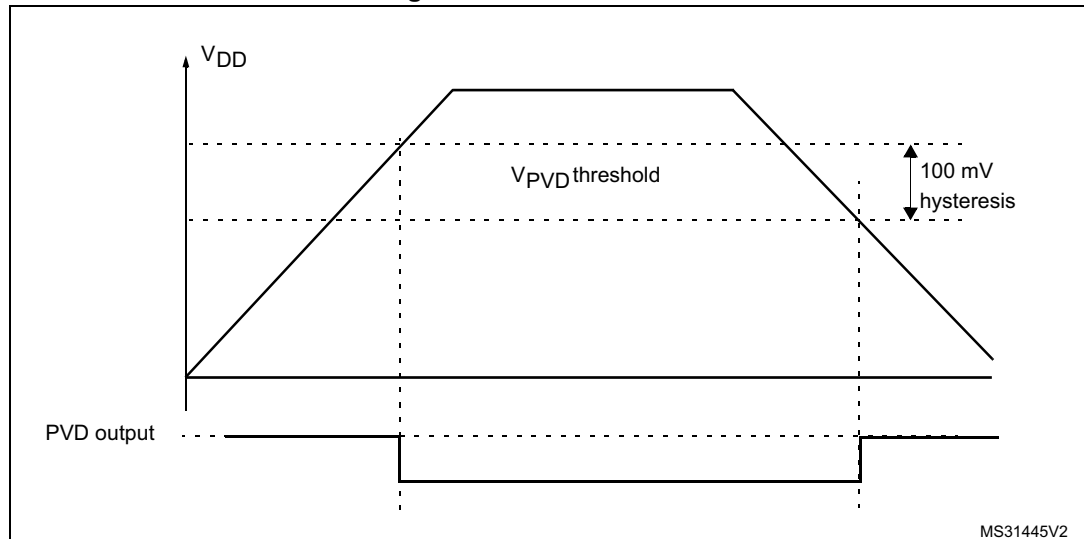
You can use the PVD to monitor the  $V_{DD}$  power supply by comparing it to a threshold selected by the PLS[2:0] bits in the [Power control register 2 \(PWR\\_CR2\)](#).

The PVD is enabled by setting the PVDE bit.

A PVDO flag is available, in the [Power status register 2 \(PWR\\_SR2\)](#), to indicate if  $V_{DD}$  is higher or lower than the PVD threshold. This event is internally connected to the EXTI line16

and can generate an interrupt if enabled through the EXTI registers. The rising/falling edge sensitivity of the EXTI Line16 should be configured according to PVD output behavior i.e. if the EXTI line 16 is configured to rising edge sensitivity, the interrupt will be generated when VDD drops below the PVD threshold. As an example the service routine could perform emergency shutdown tasks.

Figure 11. PVD thresholds



### 5.2.3 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 ( $V_{DDA}$ ,  $V_{DDIO2}$  and  $V_{DDUSB}$ ) can be independent from  $V_{DD}$  and can be monitored with four Peripheral Voltage Monitoring (PVM).

Each of the four PVMx ( $x=1, 2, 3, 4$ ) is a comparator between a fixed threshold  $V_{PVMx}$  and the selected power supply. PVM0x flags indicate if the independent power supply is higher or lower than the PVMx threshold: PVM0x 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 Stop 0, Stop 1 and Stop 2 modes, and the PVM interrupt can wake up from the Stop mode.

Table 22. PVM features

PVM	Power supply	PVM threshold	EXTI line
PVM1	$V_{DDUSB}$	$V_{PVM1}$ (around 1.2 V)	35
PVM2	$V_{DDIO2}$	$V_{PVM2}$ (around 0.9 V)	36
PVM3	$V_{DDA}$	$V_{PVM3}$ (around 1.65 V)	37
PVM4	$V_{DDA}$	$V_{PVM4}$ (around 2.2 V)	38

The independent supplies ( $V_{DDA}$ ,  $V_{DDIO2}$  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 supplies.

- If these supplies are shorted externally to  $V_{DD}$ , the application should assume they are available without enabling any Peripheral Voltage Monitoring.
- If these supplies are independent from  $V_{DD}$ , the Peripheral Voltage Monitoring (PVM) can be enabled to confirm whether the supply is present or not.

The following sequence must be done before using the USB OTG peripheral:

1. If  $V_{DDUSB}$  is independent from  $V_{DD}$ :
  - a) Enable the PVM1 by setting PVME1 bit in the *Power control register 2 (PWR\_CR2)*.
  - b) Wait for the PVM1 wakeup time
  - c) Wait until PVMO1 bit is cleared in the *Power status register 2 (PWR\_SR2)*.
  - d) Optional: Disable the PVM1 for consumption saving.
2. Set the USV bit in the *Power control register 2 (PWR\_CR2)* to remove the  $V_{DDUSB}$  power isolation.

The following sequence must be done before using any I/O from PG[15:2]:

1. If  $V_{DDIO2}$  is independent from  $V_{DD}$ :
  - a) Enable the PVM2 by setting PVME2 bit in the *Power control register 2 (PWR\_CR2)*.
  - b) Wait for the PVM2 wakeup time
  - c) Wait until PVMO2 bit is cleared in the *Power status register 2 (PWR\_SR2)*.
  - d) Optional: Disable the PVM2 for consumption saving.
2. Set the IOSV bit in the *Power control register 2 (PWR\_CR2)* to remove the  $V_{DDIO2}$  power isolation.

The following sequence must be done before using any of these analog peripherals: 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 (or PVM4) by setting PVME3 (or PVME4) bit in the *Power control register 2 (PWR\_CR2)*.
  - b) Wait for the PVM3 (or PVM4) wakeup time
  - c) Wait until PVMO3 (or PVMO4) bit is cleared in the *Power status register 2 (PWR\_SR2)*.
  - d) Optional: Disable the PVM3 (or PVM4) for consumption saving.
2. Enable the analog peripheral, which automatically removes the  $V_{DDA}$  isolation.

### 5.3 Low-power modes

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

The device features seven low-power modes:

- Sleep mode: CPU clock off, all peripherals including Cortex<sup>®</sup>-M4 core peripherals such as NVIC, SysTick, etc. can run and wake up the CPU when an interrupt or an event occurs. Refer to [Section 5.3.4: Sleep mode](#).
- Low-power run mode: This mode is achieved when the system clock frequency is reduced below 2 MHz. The code is executed from the SRAM or the Flash memory. The regulator is in low-power mode to minimize the regulator's operating current. Refer to [Section 5.3.2: Low-power run mode \(LP run\)](#).
- Low-power sleep mode: This mode is entered from the Low-power run mode: Cortex<sup>®</sup>-M4 is off. Refer to [Section 5.3.5: Low-power sleep mode \(LP sleep\)](#).
- Stop 0, Stop 1 and Stop 2 modes: SRAM1, SRAM2, SRAM3 and all registers content are 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 the Stop mode to detect their wakeup condition.

In Stop 2 mode, most of the  $V_{CORE}$  domain is put in a lower leakage mode. To further reduce the current consumption during Stop 2, it is possible to switch OFF the SRAM3:

- Stop 2 mode with SRAM3 content lost when the RRSTP bit is cleared in PWR\_CR1 register (default setting).

Stop 2 mode with SRAM3 retention when the RRSTP bit is set in PWR\_CR1 register.

Stop 1 offers the largest number of active peripherals and wakeup sources, a smaller wakeup time but a higher consumption than Stop 2. In Stop 0 mode, the main regulator remains ON, which allows the fastest wakeup time but with much higher consumption. The active peripherals and wakeup sources are the same as in Stop 1 mode.

The system clock, when exiting from Stop 0, Stop 1 or Stop 2 mode, can be either MSI up to 48 MHz or HSI16, depending on the software configuration.

Refer to [Section 5.3.6: Stop 0 mode](#) and [Section 5.3.8: Stop 2 mode](#).

- Standby mode:  $V_{CORE}$  domain is powered off. However, it is possible to preserve the SRAM2 contents:
  - Standby mode with SRAM2 retention when the bit RRS is set in PWR\_CR3 register. In this case, SRAM2 is supplied by the low-power regulator.
  - Standby mode when the bit RRS is cleared in PWR\_CR3 register. In this case the main regulator and the low-power regulator are powered off.

All clocks in the  $V_{CORE}$  domain are stopped, 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 MSI from 1 MHz up to 8 MHz.

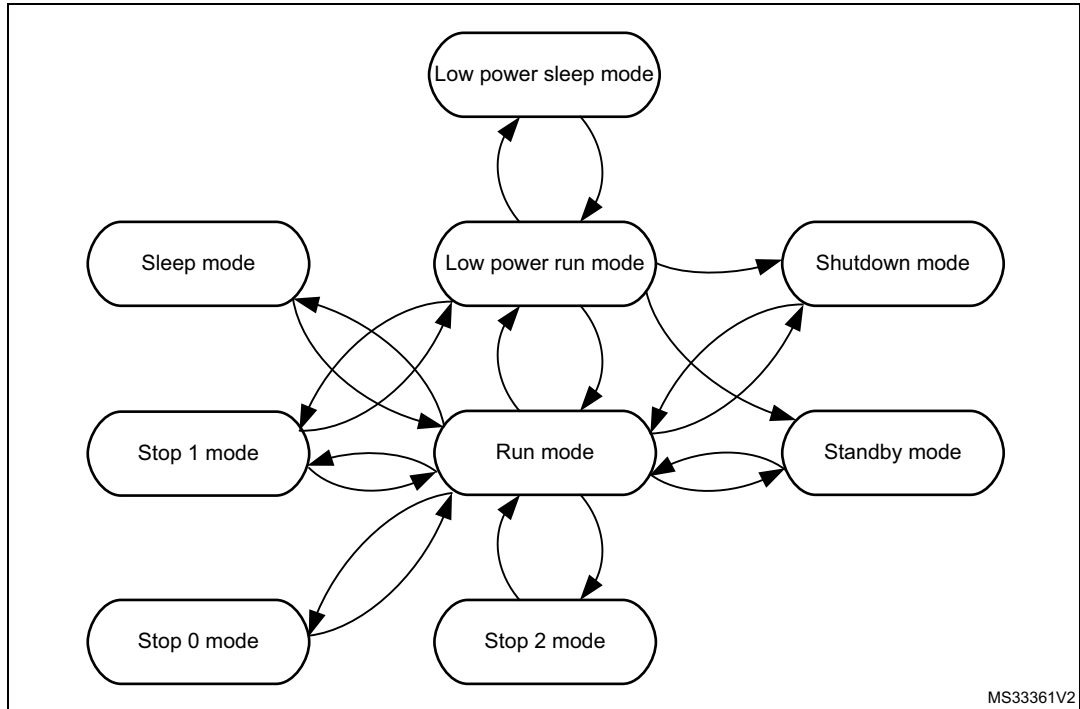
Refer to [Section 5.3.9: Standby mode](#).

- 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 the 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. Refer to [Section 5.3.10: Shutdown mode](#).

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

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

Figure 12. Low-power modes possible transitions



**Table 23. Low-power mode summary**

Mode name	Entry	Wakeup source <sup>(1)</sup>	Wakeup system clock	Effect on clocks	Voltage regulators	
					MR	LPR
Sleep (Sleep-now or Sleep-on-exit)	WFI or Return from ISR	Any interrupt	Same as before entering Sleep mode	CPU clock OFF no effect on other clocks or analog clock sources	ON	ON
	WFE	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	OFF	ON
	Set LPR bit + WFE	Wakeup event				
Stop 0	LPMS="000" + SLEEPDEEP bit + WFI or Return from ISR or WFE	Any EXTI line (configured in the EXTI registers) Specific peripherals events	HSI16 when STOPWUCK=1 in RCC_CFGR MSI with the frequency before entering the Stop mode when STOPWUCK=0.	All clocks OFF except LSI and LSE	ON	ON
Stop 1	LPMS="001" + SLEEPDEEP bit + WFI or Return from ISR or WFE					
Stop 2	LPMS="010" + SLEEPDEEP bit + WFI or Return from ISR or WFE					
Standby with SRAM2	LPMS="011" + Set RRS bit + SLEEPDEEP bit + WFI or Return from ISR or WFE	WKUP pin edge, RTC event, external reset in NRST pin, IWDG reset	MSI from 1 MHz up to 8 MHz		OFF	OFF
Standby	LPMS="011" + Clear RRS bit + SLEEPDEEP bit + WFI or Return from ISR or WFE	WKUP pin edge, RTC event, 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 24: Functionalities depending on the working mode.](#)



Table 24. Functionalities depending on the working mode<sup>(1)</sup>

Peripheral	Run	Sleep	Low-power run	Low-power sleep	Stop 0/1		Stop 2		Standby		Shutdown		VBAT
					-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	
CPU	Y	-	Y	-	-	-	-	-	-	-	-	-	-
Flash memory (2 MB)	O <sup>(2)</sup>	O <sup>(2)</sup>	O <sup>(2)</sup>	O <sup>(2)</sup>	-	-	-	-	-	-	-	-	-
SRAM1 (192 KB)	Y	Y <sup>(3)</sup>	Y	Y <sup>(3)</sup>	Y	-	Y	-	-	-	-	-	-
SRAM2 (64 KB)	Y	Y <sup>(3)</sup>	Y	Y <sup>(3)</sup>	Y	-	Y	-	O <sup>(4)</sup>	-	-	-	-
SRAM3 (384 KB)	Y	Y <sup>(3)</sup>	Y	Y <sup>(3)</sup>	Y	-	Y	-	-	-	-	-	-
FSMC	O	O	O	O	-	-	-	-	-	-	-	-	-
OCTOSPI	O	O	O	O	-	-	-	-	-	-	-	-	-
Backup Registers	Y	Y	Y	Y	Y	-	Y	-	Y	-	Y	-	Y
Brown-out reset (BOR)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-
Programmable Voltage Detector (PVD)	O	O	O	O	O	O	O	O	-	-	-	-	-
Peripheral Voltage Monitor (PVMx; x=1,2,3,4)	O	O	O	O	O	O	O	O	-	-	-	-	-
DMA	O	O	O	O	-	-	-	-	-	-	-	-	-
DMA2D	O	O	O	O	-	-	-	-	-	-	-	-	-
Oscillator HSI16	O	O	O	O	(5)	-	(5)	-	-	-	-	-	-
Oscillator HSI48	O	O	-	-	-	-	-	-	-	-	-	-	-
High Speed External (HSE)	O	O	O	O	-	-	-	-	-	-	-	-	-
Low Speed Internal (LSI)	O	O	O	O	O	-	O	-	O	-	-	-	-
Low Speed External (LSE)	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	-	-	-
RTC / Auto wakeup	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
Camera interface	O	O	O	O	-	-	-	-	-	-	-	-	-

Table 24. Functionalities depending on the working mode<sup>(1)</sup> (continued)

Peripheral	Run	Sleep	Low-power run	Low-power sleep	Stop 0/1		Stop 2		Standby		Shutdown		VBAT
					-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	
LCD-TFT	O	O	-	-	-	-	-	-	-	-	-	-	-
GFXMMU	Y	Y <sup>(3)</sup>	Y	Y <sup>(3)</sup>	Y	-	Y	-	-	-	-	-	-
DSIHOST	O	O	-	-	-	-	-	-	-	-	-	-	-
USB OTG FS	O <sup>(8)</sup>	O <sup>(8)</sup>	-	-	-	O	-	-	-	-	-	-	-
USARTx (x=1,2,3,4,5)	O	O	O	O	O <sup>(6)</sup>	O <sup>(6)</sup>	-	-	-	-	-	-	-
Low-power UART (LPUART1)	O	O	O	O	O <sup>(6)</sup>	O <sup>(6)</sup>	O <sup>(6)</sup>	O <sup>(6)</sup>	-	-	-	-	-
I2Cx (x=1,2,4)	O	O	O	O	O <sup>(7)</sup>	O <sup>(7)</sup>	-	-	-	-	-	-	-
I2C3	O	O	O	O	O <sup>(7)</sup>	O <sup>(7)</sup>	O <sup>(7)</sup>	O <sup>(7)</sup>	-	-	-	-	-
SPIx (x=1,2,3)	O	O	O	O	-	-	-	-	-	-	-	-	-
CAN1	O	O	O	O	-	-	-	-	-	-	-	-	-
SDMMC1	O	O	O	O	-	-	-	-	-	-	-	-	-
SAIx (x=1,2)	O	O	O	O	-	-	-	-	-	-	-	-	-
DFSDM1	O	O	O	O	-	-	-	-	-	-	-	-	-
ADC1	O	O	O	O	-	-	-	-	-	-	-	-	-
DACx (x=1,2)	O	O	O	O	O	-	-	-	-	-	-	-	-
VREFBUF	O	O	O	O	O	-	-	-	-	-	-	-	-
OPAMPx (x=1,2)	O	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	-	-	-	-	-	-	-
Independent watchdog (IWDG)	O	O	O	O	O	O	O	O	O	O	-	-	-
Window watchdog (WWDG)	O	O	O	O	-	-	-	-	-	-	-	-	-
SysTick timer	O	O	O	O	-	-	-	-	-	-	-	-	-

**Table 24. Functionalities depending on the working mode<sup>(1)</sup> (continued)**

Peripheral	Run	Sleep	Low-power run	Low-power sleep	Stop 0/1		Stop 2		Standby		Shutdown		VBAT
					-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	
Touch sensing controller (TSC)	O	O	O	O	-	-	-	-	-	-	-	-	-
Random number generator (RNG)	O <sup>(8)</sup>	O <sup>(8)</sup>	-	-	-	-	-	-	-	-	-	-	-
AES hardware accelerator	O	O	O	O	-	-	-	-	-	-	-	-	-
HASH hardware accelerator	O	O	O	O	-	-	-	-	-	-	-	-	-
CRC calculation unit	O	O	O	O	-	-	-	-	-	-	-	-	-
GPIOs	O	O	O	O	O	O	O	O	<sup>(9)</sup>	5 pins <sup>(10)</sup>	<sup>(11)</sup>	5 pins <sup>(10)</sup>	-

- Legend: Y = Yes (Enable). O = Optional (Disable by default. Can be enabled by software). - = Not available.
- The Flash can be configured in Power-down mode. By default, it is not in Power-down mode.
- The SRAM clock can be gated on or off.
- SRAM2 content is preserved when the bit RRS is set in PWR\_CR3 register.
- 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 which requested it. HSI16 is automatically put off when the peripheral does not need it anymore.
- UART and LPUART reception is functional in Stop mode, and generates a wakeup interrupt on Start, address match or received frame event.
- I2C address detection is functional in Stop mode, and generates a wakeup interrupt in case of address match.
- Voltage scaling Range 1 only.
- I/Os can be configured with internal pull-up, pull-down or floating in Standby mode.
- The I/Os with wakeup from Standby/Shutdown capability are: PA0, PC13, PE6, PA2, PC5.
- I/Os can be configured with internal pull-up, pull-down or floating in Shutdown mode but the configuration is lost when exiting the Shutdown mode.

### Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop 0, Stop1, Stop 2, Standby or Shutdown mode while the debug features are used. This is due to the fact that the Cortex<sup>®</sup>-M4 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 46.16.1: Debug support for low-power modes](#).

### 5.3.1 Run mode

#### Slowing down system clocks

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

For more details, refer to [Section 6.4.3: 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 the 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.

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

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

Please 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 the 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 by setting the RUN\_PD bit in the [Flash access control register \(FLASH\\_ACR\)](#).
2. Decrease the system clock frequency below 2 MHz.
3. Force the regulator in low-power mode by setting the LPR bit in the PWR\_CR1 register.

Refer to [Table 25: Low-power run](#) on how to enter the Low-power run mode.

#### Exiting the Low-power run mode

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

1. Force the regulator in main mode by clearing the LPR bit in the PWR\_CR1 register.
2. Wait until REGLPF bit is cleared in the PWR\_SR2 register.
3. Increase the system clock frequency.

Refer to [Table 25: Low-power run](#) on how to exit the Low-power run mode.

Table 25. 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

### 5.3.3 Low-power modes

#### Entering low-power mode

Low-power modes are entered by the MCU by executing the WFI (Wait For Interrupt), or WFE (Wait for Event) instructions, or when the SLEEPONEXIT bit in the Cortex<sup>®</sup>-M4 System Control register is set on Return from ISR.

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

#### Exiting low-power mode

From Sleep modes, and Stop modes the MCU exit low-power mode depending on the way the low-power 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:
  - NVIC IRQ interrupt.
    - When SEVONPEND = 0 in the Cortex<sup>®</sup>-M4 System Control register. By 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 will wakeup and interrupt the MCU.

- When SEVONPEND = 1 in the Cortex<sup>®</sup>-M4 System Control register.

By 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 will wakeup the MCU, even the disabled ones. Only enabled NVIC interrupts with sufficient priority will wakeup and interrupt the MCU.

- Event

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

It may be necessary to clear the interrupt flag in the peripheral.

From Standby modes, and Shutdown modes the MCU exit low-power mode through an external reset (NRST pin), an IWDG reset, a rising edge on one of the enabled WKUPx pins or a RTC event occurs (see [Figure 420: RTC block diagrams](#)).

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.).

### 5.3.4 Sleep mode

#### I/O states in Sleep mode

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

#### Entering the Sleep mode

The Sleep mode is entered according [Section : Entering low-power mode](#), when the SLEEPDEEP bit in the Cortex<sup>®</sup>-M4 System Control register is clear.

Refer to [Table 26: Sleep](#) for details on how to enter the Sleep mode.

#### Exiting the Sleep mode

The Sleep mode is exit according [Section : Exiting low-power mode](#).

Refer to [Table 26: Sleep](#) for more details on how to exit the Sleep mode.

**Table 26. Sleep**

Sleep-now mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: <ul style="list-style-type: none"> <li>- SLEEPDEEP = 0</li> <li>- No interrupt (for WFI) or event (for WFE) is pending</li> </ul> Refer to the Cortex <sup>®</sup> -M4 System Control register.
	On return from ISR while: <ul style="list-style-type: none"> <li>- SLEEPDEEP = 0 and</li> <li>- SLEEPONEXIT = 1</li> <li>- No interrupt is pending</li> </ul> Refer to the Cortex <sup>®</sup> -M4 System Control register.

Table 26. Sleep (continued)

Sleep-now mode	Description
Mode exit	<p>If WFI or return from ISR was used for entry Interrupt: refer to <a href="#">Table 69: STM32L4Rxxx and STM32L4Sxxx vector table</a></p> <p>If WFE was used for entry and SEVONPEND = 0: Wakeup event: refer to <a href="#">Section 16.3.2: Wakeup event management</a></p> <p>If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC: refer to <a href="#">Table 69: STM32L4Rxxx and STM32L4Sxxx vector table</a> or Wakeup event: refer to <a href="#">Section 16.3.2: Wakeup event management</a></p>
Wakeup latency	None

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

Please 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 the Low-power sleep mode

The Low-power sleep mode is entered from Low-power run mode according [Section : Entering low-power mode](#), when the SLEEPDEEP bit in the Cortex<sup>®</sup>-M4 System Control register is clear.

Refer to [Table 27: Low-power sleep](#) for details on how to enter the Low-power sleep mode.

#### Exiting the Low-power sleep mode

The low-power Sleep mode is exit according [Section : Exiting low-power mode](#). When exiting the Low-power sleep mode by issuing an interrupt or an event, the MCU is in Low-power run mode.

Refer to [Table 27: Low-power sleep](#) for details on how to exit the Low-power sleep mode.

Table 27. Low-power sleep

Low-power sleep-now mode	Description
Mode entry	Low-power sleep mode is entered from the Low-power run mode. WFI (Wait for Interrupt) or WFE (Wait for Event) while: <ul style="list-style-type: none"> <li>– SLEEPDEEP = 0</li> <li>– No interrupt (for WFI) or event (for WFE) is pending</li> </ul> Refer to the Cortex <sup>®</sup> -M4 System Control register.
	Low-power sleep mode is entered from the Low-power run mode. On return from ISR while: <ul style="list-style-type: none"> <li>– SLEEPDEEP = 0 and</li> <li>– SLEEPONEXIT = 1</li> <li>– No interrupt is pending</li> </ul> Refer to the Cortex <sup>®</sup> -M4 System Control register.
Mode exit	<p>If WFI or Return from ISR was used for entry</p> <p>Interrupt: refer to <a href="#">Table 69: STM32L4Rxxx and STM32L4Sxxx vector table</a></p> <p>If WFE was used for entry and SEVONPEND = 0:</p> <p>Wakeup event: refer to <a href="#">Section 16.3.2: Wakeup event management</a></p> <p>If WFE was used for entry and SEVONPEND = 1:</p> <p>Interrupt even when disabled in NVIC: refer to <a href="#">Table 69: STM32L4Rxxx and STM32L4Sxxx vector table</a></p> <p>Wakeup event: refer to <a href="#">Section 16.3.2: Wakeup event management</a></p> <p>After exiting the Low-power sleep mode, the MCU is in Low-power run mode.</p>
Wakeup latency	None

### 5.3.6 Stop 0 mode

The Stop 0 mode is based on the Cortex<sup>®</sup>-M4 Deepsleep mode combined with the peripheral clock gating. The voltage regulator is configured in main regulator mode. In Stop 0 mode, all clocks in the V<sub>CORE</sub> domain are stopped; the PLL, the MSI, the HSI16 and the HSE oscillators are disabled. Some peripherals with the wakeup capability (I2Cx (x=1,2,3), U(S)ARTx(x=1,2...5) 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 Stop 0 mode. The consumption is increased when thresholds higher than V<sub>BOR0</sub> are used.

#### I/O states in Stop 0 mode

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

#### Entering the Stop 0 mode

The Stop 0 mode is entered according [Section : Entering low-power mode](#), when the SLEEPDEEP bit in the Cortex<sup>®</sup>-M4 System Control register is set.



Refer to [Table 28: Stop 0 mode](#) for details on how to enter the Stop 0 mode.

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

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

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

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started, it cannot be stopped except by a Reset. See [Section 40.3: 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 LSION 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 \(RCC\\_BDCR\)](#).

Several peripherals can be used in Stop 0 mode and can add consumption if they are enabled and clocked by LSI or LSE, or when they request the HSI16 clock: LPTIM1, LPTIM2, I2Cx (x=1,2,3,4) U(S)ARTx(x=1,2...5), LPUART.

The DACx (x=1,2), the OPAMPs and the comparators can be used in Stop 0 mode, the PVMx (x=1,2,3,4) and the PVD as well. If they are not needed, they must be disabled by software to save their power consumptions.

The ADCx (x=1,2,3), temperature sensor and VREFBUF buffer can consume power during the Stop 0 mode, unless they are disabled before entering this mode.

### Exiting the Stop 0 mode

The Stop 0 mode is exit according [Section : Entering low-power mode](#).

Refer to [Table 28: Stop 0 mode](#) for details on how to exit Stop 0 mode.

When exiting Stop 0 mode by issuing an interrupt or a wakeup event, the HSI16 oscillator is selected as system clock if the bit STOPWUCK is set in [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 the voltage regulator operates in low-power mode, an additional startup delay is incurred when waking up from Stop 0 mode with HSI16. By keeping the internal regulator ON during Stop 0 mode, the consumption is higher although the startup time is reduced.

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

**Table 28. Stop 0 mode**

Stop 0 mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex<sup>®</sup>-M4 System Control register</li> <li>– No interrupt (for WFI) or event (for WFE) is pending</li> <li>– LPMS = “000” in PWR_CR1</li> </ul>
	On Return from ISR while: <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex<sup>®</sup>-M4 System Control register</li> <li>– SLEEPONEXIT = 1</li> <li>– No interrupt is pending</li> <li>– LPMS = “000” in PWR_CR1</li> </ul>
	<i>Note: To enter Stop 0 mode, all EXTI Line pending bits (in <a href="#">Pending register 1 (EXTI_PR1)</a>), and the peripheral flags generating wakeup interrupts must be cleared. Otherwise, the Stop 0 mode entry procedure is ignored and program execution continues.</i>
Mode exit	If WFI or Return from ISR was used for entry Any EXTI Line configured in interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to <a href="#">Table 69: STM32L4Rxxx and STM32L4Sxxx vector table</a> . If WFE was used for entry and SEVONPEND = 0: Any EXTI Line configured in event mode. Refer to <a href="#">Section 16.3.2: Wakeup event management</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 69: STM32L4Rxxx and STM32L4Sxxx vector table</a> . Wakeup event: refer to <a href="#">Section 16.3.2: Wakeup event management</a>
Wakeup latency	Longest wakeup time between: MSI or HSI16 wakeup time and Flash wakeup time from Stop 0 mode.

### 5.3.7 Stop 1 mode

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

Refer to [Table 29: Stop 1 mode](#) for details on how to enter and exit Stop 1 mode.

Table 29. Stop 1 mode

Stop 1 mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex<sup>®</sup>-M4 System Control register</li> <li>– No interrupt (for WFI) or event (for WFE) is pending</li> <li>– LPMS = “001” in PWR_CR1</li> </ul>
	On Return from ISR while: <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex<sup>®</sup>-M4 System Control register</li> <li>– SLEEPONEXIT = 1</li> <li>– No interrupt is pending</li> <li>– LPMS = “001” in PWR_CR1</li> </ul>
	<i>Note: To enter Stop 1 mode, all EXTI Line pending bits (in <a href="#">Pending register 1 (EXTI_PR1)</a>), and the peripheral flags generating wakeup interrupts must be cleared. Otherwise, the Stop 1 mode entry procedure is ignored and program execution continues.</i>
Mode exit	If WFI or Return from ISR was used for entry Any EXTI Line configured in interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to <a href="#">Table 69: STM32L4Rxxx and STM32L4Sxxx vector table</a> .
	If WFE was used for entry and SEVONPEND = 0: Any EXTI Line configured in event mode. Refer to <a href="#">Section 16.3.2: Wakeup event management</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 69: STM32L4Rxxx and STM32L4Sxxx vector table</a> . Wakeup event: refer to <a href="#">Section 16.3.2: Wakeup event management</a>
Wakeup latency	Longest wakeup time between: MSI or HSI16 wakeup time and regulator wakeup time from Low-power mode + Flash wakeup time from Stop 1 mode.

### 5.3.8 Stop 2 mode

The Stop 2 mode is based on the Cortex<sup>®</sup>-M4 Deepsleep mode combined with peripheral clock gating. In Stop 2 mode, all clocks in the V<sub>CORE</sub> 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, SRAM3 and register contents are preserved. The SRAM3 content is preserved or lost following the RRSTP bit configuration in the PWR\_CR1 register. By default, after reset, the RRSTP bit is reset thus the SRAM3 content is lost during Stop 2.

The BOR is always available in Stop 2 mode. The consumption is increased when thresholds higher than V<sub>BOR0</sub> are used.

**Note:** *The comparators outputs, the LPUART outputs and the LPTIM1 outputs are forced to low speed (OSPEEDy=00) during the Stop 2 mode.*

### I/O states in Stop 2 mode

In the Stop 2 mode, all I/O pins keep the same state as in the Run mode.

### Entering Stop 2 mode

The Stop 2 mode is entered according [Section : Entering low-power mode](#), when the SLEEPDEEP bit in the Cortex<sup>®</sup>-M4 System Control register is set.

Refer to [Table 30: Stop 2 mode](#) for details on how to enter the Stop 2 mode.

Stop 2 mode can only be entered from Run mode. It is not possible to enter Stop 2 mode from the Low-power run mode.

If Flash memory programming is ongoing, the Stop 2 mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, The Stop 2 mode entry is delayed until the APB access is finished.

In Stop 2 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 40.3: IWDG functional description](#) in [Section 40: Independent watchdog \(IWDG\)](#).
- Real-time clock (RTC): this is configured by the RTCEN bit in the [Backup domain control register \(RCC\\_BDCR\)](#)
- Internal RC oscillator (LSI): this is configured by the LSION bit in the [Control/status register \(RCC\\_CSR\)](#).
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the [Backup domain control register \(RCC\\_BDCR\)](#).

Several peripherals can be used in Stop 2 mode and can add consumption if they are enabled and clocked by LSI or LSE, or when they request the HSI16 clock: LPTIM1, I2C3, LPUART.

The comparators can be used in Stop 2 mode, the PVMx (x=1,2,3,4) and the PVD as well. If they are not needed, they must be disabled by software to save their power consumptions.

The ADCx, OPAMPx, DACx, temperature sensor and VREFBUF buffer can consume power during Stop 2 mode, unless they are disabled before entering this mode.

All the peripherals which cannot be enabled in Stop 2 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 [AHB1 peripheral reset register \(RCC\\_AHB1RSTR\)](#), [AHB2 peripheral reset register \(RCC\\_AHB2RSTR\)](#), [AHB3 peripheral reset register \(RCC\\_AHB3RSTR\)](#), [APB1 peripheral reset register 1 \(RCC\\_APB1RSTR1\)](#), [APB1 peripheral reset register 2 \(RCC\\_APB1RSTR2\)](#), [APB2 peripheral reset register \(RCC\\_APB2RSTR\)](#).

### Exiting Stop 2 mode

The Stop 2 mode is exit according [Section : Exiting low-power mode](#).

Refer to [Table 30: Stop 2 mode](#) for details on how to exit Stop 2 mode.

When exiting Stop 2 mode by issuing an interrupt or a wakeup event, the HSI16 oscillator is selected as system clock if the bit STOPWUCK is set in [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 the Stop 2 mode, the MCU is in Run mode (Range 1 or Range 2 depending on VOS bit in PWR\_CR1).

**Table 30. Stop 2 mode**

Stop 2 mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex<sup>®</sup>-M4 System Control register</li> <li>– No interrupt (for WFI) or event (for WFE) is pending</li> <li>– LPMS = “010” in PWR_CR1</li> </ul>
	On return from ISR while: <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex<sup>®</sup>-M4 System Control register</li> <li>– SLEEPONEXIT = 1</li> <li>– No interrupt is pending</li> <li>– LPMS = “010” in PWR_CR1</li> </ul>
	<i>Note: To enter Stop 2 mode, all EXTI Line pending bits (in <a href="#">Pending register 1 (EXTI_PR1)</a>), and the peripheral flags generating wakeup interrupts must be cleared. Otherwise, the Stop mode entry procedure is ignored and program execution continues.</i>
Mode exit	If WFI or Return from ISR was used for entry: <p>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 69: STM32L4Rxxx and STM32L4Sxxx vector table</a>.</p> If WFE was used for entry and SEVONPEND = 0: <p>Any EXTI Line configured in event mode. Refer to <a href="#">Section 16.3.2: Wakeup event management</a>.</p> If WFE was used for entry and SEVONPEND = 1: <p>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 69: STM32L4Rxxx and STM32L4Sxxx vector table</a>.</p> <p>Any EXTI Line configured in event mode. Refer to <a href="#">Section 16.3.2: Wakeup event management</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 Stop 2 mode.

### 5.3.9 Standby mode

The Standby mode allows to achieve the lowest power consumption with BOR. It is based on the Cortex<sup>®</sup>-M4 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\_CR3 register. 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 the Standby mode, the IO's are by default in floating state. If the APC bit of PWR\_CR3 register has been set, 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 if none of the PWR\_PUCRx or PWR\_PDCRx register has been set. The pull-down configuration has highest priority over pull-up configuration in case both PWR\_PUCRx and PWR\_PDCRx are set for the same IO.

Some I/Os (listed in [Section 8.3.1: General-purpose I/O \(GPIO\)](#)) are used for JTAG/SW debug and can only be configured to their respective reset pull-up or pull-down state during Standby mode setting their respective bit in the PWR\_PUCRx or PWR\_PDCRx registers to '1', or will be configured to floating state if the bit is kept at '0'.

The RTC outputs on PC13 are functional in Standby mode. PC14 and PC15 used for LSE are also functional. 5 wakeup pins (WKUPx, x=1,2...5) and the 3 RTC tamperers are available.

### Entering Standby mode

The Standby mode is entered according [Section : Entering low-power mode](#), when the SLEEPDEEP bit in the Cortex<sup>®</sup>-M4 System Control register is set.

Refer to [Table 31: Standby mode](#) 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 40.3: IWDG functional description](#) in [Section 40: Independent watchdog \(IWDG\)](#).
- real-time clock (RTC): this is configured by the RTCEN bit in the Backup domain control register (RCC\_BDCR)
- Internal RC oscillator (LSI): this is configured by the LSION bit in the Control/status register (RCC\_CSR).
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the Backup domain control register (RCC\_BDCR)

### Exiting Standby mode

The Standby mode is exited according [Section : Entering low-power mode](#). The SBF status flag in the [Power control register 3 \(PWR\\_CR3\)](#) indicates that the MCU was in Standby mode. All registers are reset after wakeup from Standby except for [Power control register 3 \(PWR\\_CR3\)](#).

Refer to [Table 31: Standby mode](#) for more details on how to exit Standby mode.

When exiting Standby mode, I/O's that were configured with pull-up or pull-down during Standby through registers PWR\_PUCRx or PWR\_PDCRx will keep this configuration upon exiting Standby mode until the bit APC of PWR\_CR3 register has been cleared. Once the bit APC is cleared, they will be either configured to their reset values or to the pull-up/pull-down state according the GPIOx\_PUPDR registers. The content of the PWR\_PUCRx or PWR\_PDCRx registers however is not lost and can be re-used for a sub-sequent entering into Standby mode.

Some I/Os (listed in [Section 8.3.1: General-purpose I/O \(GPIO\)](#)) are used for JTAG/SW debug and have internal pull-up or pull-down activated after reset so will be configured at this reset value as well when exiting Standby mode.

For IO's, with a pull-up or pull-down pre-defined after reset (some JTAG/SW IO's) or with GPIOx\_PUPDR programming done after exiting from Standby, in case those programming is different from the PWR\_PUCRx or PWR\_PDCRx programmed value during Standby, both a pull-down and pull-up will be applied until the bit APC is cleared, releasing the PWR\_PUCRx or PWR\_PDCRx programmed value.

**Table 31. Standby mode**

Standby mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex®-M4 System Control register</li> <li>– No interrupt (for WFI) or event (for WFE) is pending</li> <li>– LPMS = “011” in PWR_CR1</li> <li>– WUFx bits are cleared in power status register 1 (PWR_SR1)</li> </ul>
	On return from ISR while: <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex®-M4 System Control register</li> <li>– SLEEPONEXIT = 1</li> <li>– No interrupt is pending</li> <li>– LPMS = “011” in PWR_CR1 and</li> <li>– WUFx bits are cleared in power status register 1 (PWR_SR1)</li> <li>– The RTC flag corresponding to the chosen wakeup source (RTC Alarm A, RTC Alarm B, RTC wakeup, tamper or timestamp flags) is cleared</li> </ul>
Mode exit	WKUPx pin edge, RTC event, external Reset in NRST pin, IWDG Reset, BOR reset
Wakeup latency	Reset phase



### 5.3.10 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, SRAM3 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, are by default in floating state. If the APC bit of PWR\_CR3 register has been set, 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 if none of the PWR\_PUCRx or PWR\_PDCRx register has been set. The pull-down configuration has highest priority over pull-up configuration in case both PWR\_PUCRx and PWR\_PDCRx are set for the same IO. However this configuration is lost when exiting the Shutdown mode due to the power-on reset.

Some I/Os (listed in [Section 8.3.1: General-purpose I/O \(GPIO\)](#)) are used for JTAG/SW debug and can only be configured to their respective reset pull-up or pull-down state during Standby mode setting their respective bit in the PWR\_PUCRx or PWR\_PDCRx registers to '1', or will be configured to floating state if the bit is kept at '0'.

The RTC outputs on PC13 are functional in Shutdown mode. PC14 and PC15 used for LSE are also functional. 5 wakeup pins (WKUPx, x=1,2...5) and the 3 RTC tampers are available.

#### Entering Shutdown mode

The Shutdown mode is entered according [Entering low-power mode](#), when the SLEEPDEEP bit in the Cortex<sup>®</sup>-M4 System Control register is set.

Refer to [Table 32: Shutdown mode](#) 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 Backup domain control register (RCC\_BDCR). Caution: in case of VDD power-down the RTC content will be lost.
- external 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the Backup domain control register (RCC\_BDCR)

#### Exiting Shutdown mode

The Shutdown mode is exit according [Section : 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 32: Shutdown mode](#) for more details on how to exit Shutdown mode.

When exiting Shutdown mode, I/Os that were configured with pull-up or pull-down during Shutdown through registers PWR\_PUCRx or PWR\_PDCRx will lose their configuration and



will be configured in floating state or to their pull-up pull-down reset value (for some I/Os listed in [Section 8.3.1: General-purpose I/O \(GPIO\)](#)).

**Table 32. Shutdown mode**

Shutdown mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex<sup>®</sup>-M4 System Control register</li> <li>– No interrupt (for WFI) or event (for WFE) is pending</li> <li>– LPMS = “1XX” in PWR_CR1</li> <li>– WUFx bits are cleared in power status register 1 (PWR_SR1)</li> </ul>
	On return from ISR while: <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex<sup>®</sup>-M4 System Control register</li> <li>– SLEEPONEXT = 1</li> <li>– No interrupt is pending</li> <li>– LPMS = “1XX” in PWR_CR1 and</li> <li>– WUFx bits are cleared in power status register 1 (PWR_SR1)</li> <li>– The RTC flag corresponding to the chosen wakeup source (RTC Alarm A, RTC Alarm B, RTC wakeup, tamper or timestamp flags) is cleared</li> </ul>
Mode exit	WKUPx pin edge, RTC event, external Reset in NRST pin
Wakeup latency	Reset phase

### 5.3.11 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 [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 clock source has the advantage of saving the cost of the 32.768 kHz crystal. This internal RC Oscillator is designed to add minimum power consumption.

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

- Configure the EXTI Line 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.

## 5.4 PWR registers

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

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

Address offset: 0x00

Reset value: 0x0000 0200. This register is reset after wakeup from Standby mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LPR	Res.	Res.	Res.	VOS[1:0]		DBP	Res.	Res.	Res.	RRSTP	Res.	LPMS[2:0]		
	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: Stop 2 mode cannot be entered when LPR bit is set. Stop 1 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 7:5 Reserved, must be kept at reset value.

Bit 4 **RRSTP**: SRAM3 retention in Stop 2 mode  
 0: SRAM3 is powered off in Stop 2 mode (SRAM3 content is lost)  
 1: SRAM3 is powered in Stop 2 mode (RAM3 content is kept).

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **LPMS[2:0]**: Low-power mode selection  
 These bits select the low-power mode entered when CPU enters the Deepsleep mode.  
 000: Stop 0 mode  
 001: Stop 1 mode  
 010: Stop 2 mode  
 011: Standby mode  
 1xx: Shutdown mode  
*Note: If LPR bit is set, Stop 2 mode cannot be selected and Stop 1 mode shall be entered instead of Stop 2.*  
*In Standby mode, SRAM2 can be preserved or not, depending on RRS bit configuration in PWR\_CR3.*

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

Address offset: 0x04

Reset value: 0x0000 0000. This register is reset when exiting the Standby mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	USV	IOSV	Res.	PVME4	PVME3	PVME2	PVME1	PLS[2:0]			PVDE
					rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 10 **USV**: V<sub>DDUSB</sub> USB supply valid  
 This bit is used to validate the V<sub>DDUSB</sub> supply for electrical and logical isolation purpose. Setting this bit is mandatory to use the USB OTG\_FS peripheral. If V<sub>DDUSB</sub> is not always present in the application, the PVM can be used to determine whether this supply is ready or not.  
 0: V<sub>DDUSB</sub> is not present. Logical and electrical isolation is applied to ignore this supply.  
 1: V<sub>DDUSB</sub> is valid.

Bit 9 **IOSV**: V<sub>DDIO2</sub> Independent I/Os supply valid  
 This bit is used to validate the V<sub>DDIO2</sub> supply for electrical and logical isolation purpose. Setting this bit is mandatory to use PG[15:2]. If V<sub>DDIO2</sub> is not always present in the application, the PVM can be used to determine whether this supply is ready or not.  
 0: V<sub>DDIO2</sub> is not present. Logical and electrical isolation is applied to ignore this supply.  
 1: V<sub>DDIO2</sub> is valid.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **PVME4**: Peripheral voltage monitoring 4 enable: V<sub>DDA</sub> vs. 2.2V  
 0: PVM4 (V<sub>DDA</sub> monitoring vs. 2.2V threshold) disable.  
 1: PVM4 (V<sub>DDA</sub> monitoring vs. 2.2V threshold) enable.

Bit 6 **PVME3**: Peripheral voltage monitoring 3 enable:  $V_{DDA}$  vs. 1.62V  
 0: PVM3 ( $V_{DDA}$  monitoring vs. 1.62V threshold) disable.  
 1: PVM3 ( $V_{DDA}$  monitoring vs. 1.62V threshold) enable.

Bit 5 **PVME2**: Peripheral voltage monitoring 2 enable:  $V_{DDIO2}$  vs. 0.9V  
 0: PVM2 ( $V_{DDIO2}$  monitoring vs. 0.9V threshold) disable.  
 1: PVM2 ( $V_{DDIO2}$  monitoring vs. 0.9V threshold) enable.

Bit 4 **PVME1**: Peripheral voltage monitoring 1 enable:  $V_{DDUSB}$  vs. 1.2V  
 0: PVM1 ( $V_{DDUSB}$  monitoring vs. 1.2V threshold) disable.  
 1: PVM1 ( $V_{DDUSB}$  monitoring vs. 1.2V threshold) enable.

Bits 3:1 **PLS[2:0]**: Power voltage detector level selection.  
 These bits select the voltage threshold detected by the power voltage detector:  
 000:  $V_{PVD0}$  around 2.0 V  
 001:  $V_{PVD1}$  around 2.2 V  
 010:  $V_{PVD2}$  around 2.4 V  
 011:  $V_{PVD3}$  around 2.5 V  
 100:  $V_{PVD4}$  around 2.6 V  
 101:  $V_{PVD5}$  around 2.8 V  
 110:  $V_{PVD6}$  around 2.9 V  
 111: External input analog voltage PVD\_IN (compared internally to VREFINT)  
*Note: These bits are write-protected when the bit PVDL (PVD Lock) is set in the SYSCFG\_CBR register.*  
*These bits are reset only by a system reset.*

Bit 0 **PVDE**: Power voltage detector enable  
 0: Power voltage detector disable.  
 1: Power voltage detector enable.  
*Note: This bit is write-protected when the bit PVDL (PVD Lock) is set in the SYSCFG\_CBR register.*  
*This bit is reset only by a system reset.*

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

Address offset: 0x08

Reset value: 0x0000 8000. This register is not reset when exiting Standby modes and with the PWRRST bit in the RCC\_APB1RSTR1 register.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIWUL	Res.	Res.	DSIPD EN	Res.	APC	Res.	RRS	Res.	Res.	Res.	EWUP 5	EWUP 4	EWUP 3	EWUP 2	EWUP 1
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

0: Internal wakeup line disable.

1: Internal wakeup line enable.

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

Bit 12 **DSIPDEN**: Enable Pull-down activation on DSI pins

1: Pull-Down is enabled on DSI pins.

0: Pull-Down is disabled on DSI pins.

Bit 11 Reserved, must be kept at reset value.

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

When this bit is set, the I/O pull-up and pull-down configurations defined in the PWR\_PUCRx and PWR\_PDCRx registers are applied. When this bit is cleared, the PWR\_PUCRx and PWR\_PDCRx registers are not applied to the I/Os, instead the I/Os will be in floating mode during Standby or configured according GPIO controller GPIOx\_PUPDR register during Run mode.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **RRS**: SRAM2 retention in Standby mode

0: SRAM2 is powered off in Standby mode (SRAM2 content is lost).

1: SRAM2 is powered by the low-power regulator in Standby mode (SRAM2 content is kept).

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **EWUP5**: Enable Wakeup pin WKUP5

When this bit is set, the external wakeup pin WKUP5 is enabled and triggers a wakeup from Standby or Shutdown event when a rising or a falling edge occurs. The active edge is configured via the WP5 bit in the PWR\_CR4 register.

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

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

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

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

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

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

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

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

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

Address offset: 0x0C

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with the PWRRST bit in the RCC\_APB1RSTR1 register.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	VBR5	VBE	Res.	Res.	Res.	WP5	WP4	WP3	WP2	WP1
						rw	rw				rw	rw	rw	rw	rw

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

Bit 9 **VBR5**: V<sub>BAT</sub> battery charging resistor selection  
 0: Charge V<sub>BAT</sub> through a 5 kOhms resistor  
 1: Charge V<sub>BAT</sub> through a 1.5 kOhms resistor

Bit 8 **VBE**: V<sub>BAT</sub> battery charging enable  
 0: V<sub>BAT</sub> battery charging disable  
 1: V<sub>BAT</sub> battery charging enable

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **WP5**: Wakeup pin WKUP5 polarity  
 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)

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  
 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)

Bit 1 **WP2**: Wakeup pin WKUP2 polarity  
 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)

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)

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

Address offset: 0x10

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with the PWRRST bit in the RCC\_APB1RSTR1 register.

Access: 2 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	Res.	Res.	Res.	Res.	Res.	Res.	SBF	Res.	Res.	Res.	WUF5	WUF4	WUF3	WUF2	WUF1
r							r				r	r	r	r	r

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

Bit 15 **WUFI**: Wakeup flag internal

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

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

Bit 8 **SBF**: Standby flag

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

0: The device did not enter the Standby mode

1: The device entered the Standby mode

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **WUF5**: Wakeup flag 5

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.

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\_SCR register.

Bit 2 **WUF3**: Wakeup flag 3

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\_SCR register.

Bit 1 **WUF2**: Wakeup flag 2

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\_SCR register.

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\_SCR register.

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

Address offset: 0x14

Reset value: 0x0000 0000. 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
PVMO4	PVMO3	PVMO2	PVMO1	PVDO	VOSF	REGLP F	REGLP S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								

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

Bit 15 **PVMO4**: Peripheral voltage monitoring output:  $V_{DDA}$  vs. 2.2 V

0:  $V_{DDA}$  voltage is above PVM4 threshold (around 2.2 V).

1:  $V_{DDA}$  voltage is below PVM4 threshold (around 2.2 V).

*Note: PVMO4 is cleared when PVM4 is disabled (PVME4 = 0). After enabling PVM4, the PVM4 output is valid after the PVM4 wakeup time.*

Bit 14 **PVMO3**: Peripheral voltage monitoring output:  $V_{DDA}$  vs. 1.62 V

0:  $V_{DDA}$  voltage is above PVM3 threshold (around 1.62 V).

1:  $V_{DDA}$  voltage is below PVM3 threshold (around 1.62 V).

*Note: PVMO3 is cleared when PVM3 is disabled (PVME3 = 0). After enabling PVM3, the PVM3 output is valid after the PVM3 wakeup time.*

Bit 13 **PVMO2**: Peripheral voltage monitoring output:  $V_{DDIO2}$  vs. 0.9 V

0:  $V_{DDIO2}$  voltage is above PVM2 threshold (around 0.9 V).

1:  $V_{DDIO2}$  voltage is below PVM2 threshold (around 0.9 V).

*Note: PVMO2 is cleared when PVM2 is disabled (PVME2 = 0). After enabling PVM2, the PVM2 output is valid after the PVM2 wakeup time.*

Bit 12 **PVMO1**: Peripheral voltage monitoring output:  $V_{DDUSB}$  vs. 1.2 V

0:  $V_{DDUSB}$  voltage is above PVM1 threshold (around 1.2 V).

1:  $V_{DDUSB}$  voltage is below PVM1 threshold (around 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**: Power voltage detector output

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

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

Bit 10 **VOSF**: Voltage scaling flag

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

0: The regulator is ready in the selected voltage range

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



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

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

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

Address offset: 0x18

Reset value: 0x0000 0000.

Access: 3 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.	Res.	Res.	Res.	Res.	Res.	Res.	CSBF	Res.	Res.	Res.	CWUF 5	CWUF 4	CWUF 3	CWUF 2	CWUF 1
							w				w	w	w	w	w

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

Bit 8 **CSBF**: Clear standby flag

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

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CWUF5**: Clear wakeup flag 5

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

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

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

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

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

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

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

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

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

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

Address offset: 0x20.

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. a standard APB access (3 for a write and 2 for a read).

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

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

Bit 15 **PU15**: Port A pull-up bit 15

When set, this bit activates the pull-up on PA[15] when APC bit is set in PWR\_CR3 register. If the corresponding PD15 bit is also set, the pull-up is not activated and the pull-down is activated instead with highest priority.

Bit 14 Reserved, must be kept at reset value.

Bits 13:0 **PUy**: Port A pull-up bit y (y=0..13)

When set, this bit activates the pull-up on PA[y] when APC bit is set in PWR\_CR3 register. The pull-up is not activated if the corresponding PDy bit is also set.

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

Address offset: 0x24.

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. a standard APB access (3 for a write and 2 for a read).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PD14	Res.	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 14 **PD14**: Port A pull-down bit 14

When set, this bit activates the pull-down on PA[14] when APC bit is set in PWR\_CR3 register.

Bit 13 Reserved, must be kept at reset value.

Bits 12:0 **PDy**: Port A pull-down bit y (y=0..12)

When set, this bit activates the pull-down on PA[y] when APC bit is set in PWR\_CR3 register.

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

Address offset: 0x28.

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. a standard APB access (3 for a write and 2 for a read).

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

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

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

When set, this bit activates the pull-up on PB[y] when APC bit is set in PWR\_CR3 register. The pull-up is not activated if the corresponding PDy bit is also set.

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

Address offset: 0x2C.

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. a standard APB access (3 for a write and 2 for a read).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	Res.	PD3	PD2	PD1	PD0
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 15:5 **PDy**: Port B pull-down bit y (y=5..15)

When set, this bit activates the pull-down on PB[y] when APC bit is set in PWR\_CR3 register.

Bit 4 Reserved, must be kept at reset value.

Bits 3:0 **PDy**: Port B pull-down bit y (y=0..3)

When set, this bit activates the pull-down on PB[y] when APC bit is set in PWR\_CR3 register.

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

Address offset: 0x30.

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. a standard APB access (3 for a write and 2 for a read).

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

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

Bits 15:0 **PUy**: Port C pull-up bit y (y=0..15)

When set, this bit activates the pull-up on PC[y] when APC bit is set in PWR\_CR3 register. The pull-up is not activated if the corresponding PDy bit is also set.

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

Address offset: 0x34.

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. a standard APB access (3 for a write and 2 for a read).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

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

Bits 15:0 **PDy**: Port C pull-down bit y (y=0..15)

When set, this bit activates the pull-down on PC[y] when APC bit is set in PWR\_CR3 register.

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

Address offset: 0x38.

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. a standard APB access (3 for a write and 2 for a read).

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

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

Bits 15:0 **PUy**: Port D pull-up bit y (y=0..15)

When set, this bit activates the pull-up on PD[y] when APC bit is set in PWR\_CR3 register. The pull-up is not activated if the corresponding PDy bit is also set.

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

Address offset: 0x3C.

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. a standard APB access (3 for a write and 2 for a read).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

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

Bits 15:0 **PDy**: Port D pull-down bit y (y=0..15)

When set, this bit activates the pull-down on PD[y] when APC bit is set in PWR\_CR3 register.

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

Address offset: 0x20.

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. a standard APB access (3 for a write and 2 for a read).

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

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

Bits 15:0 **PUy**: Port E pull-up bit y (y=0..15)

When set, this bit activates the pull-up on PE[y] when APC bit is set in PWR\_CR3 register. The pull-up is not activated if the corresponding PDy bit is also set.

### 5.4.17 Power Port E pull-down control register (PWR\_PDCRE)

Address offset: 0x44.

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. a standard APB access (3 for a write and 2 for a read).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **PDy**: Port E pull-down bit y (y=0..15)

When set, this bit activates the pull-down on PE[y] when APC bit is set in PWR\_CR3 register.

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

Address offset: 0x48.

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. a standard APB access (3 for a write and 2 for a read).

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

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

Bits 15:0 **PUy**: Port F pull-up bit y (y=0..15)

When set, this bit activates the pull-up on PF[y] when APC bit is set in PWR\_CR3 register. The pull-up is not activated if the corresponding PDy bit is also set.

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

Address offset: 0x4C.

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. a standard APB access (3 for a write and 2 for a read).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

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

Bits 15:0 **PDy**: Port F pull-down bit y (y=0..15)

When set, this bit activates the pull-down on PF[y] when APC bit is set in PWR\_CR3 register.

### 5.4.20 Power Port G pull-up control register (PWR\_PUCRG)

Address offset: 0x50.

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. a standard APB access (3 for a write and 2 for a read).

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

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

Bits 15:0 **PUy**: Port G pull-up bit y (y=0..15)

When set, this bit activates the pull-up on PG[y] when APC bit is set in PWR\_CR3 register. The pull-up is not activated if the corresponding PDy bit is also set.

### 5.4.21 Power Port G pull-down control register (PWR\_PDCRG)

Address offset: 0x54.

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. a standard APB access (3 for a write and 2 for a read).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

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

Bits 15:0 **PDy**: Port G pull-down bit y (y=0..15)

When set, this bit activates the pull-down on PG[y] when APC bit is set in PWR\_CR3 register.

### 5.4.22 Power Port H pull-up control register (PWR\_PUCRH)

Address offset: 0x58.

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. a standard APB access (3 for a write and 2 for a read).



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

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

Bits 15:0 **PUy**: Port H pull-up bit y (y=0..15)

When set, this bit activates the pull-up on PH[y] when APC bit is set in PWR\_CR3 register.  
The pull-up is not activated if the corresponding PDy bit is also set.

### 5.4.23 Power Port H pull-down control register (PWR\_PDCRH)

Address offset: 0x5C.

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. a standard APB access (3 for a write and 2 for a read).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **PDy**: Port H pull-down bit y (y=0..15)

When set, this bit activates the pull-down on PH[y] when APC bit is set in PWR\_CR3 register.

### 5.4.24 Power Port I pull-up control register (PWR\_PUCRI)

Address offset: 0x60.

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. a standard APB access (3 for a write and 2 for a read).

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

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

Bits 11:0 **PUy**: Port I pull-up bit y (y=0..11)

When set, this bit activates the pull-up on PI[y] when APC bit is set in PWR\_CR3 register. The pull-up is not activated if the corresponding PDy bit is also set.

### 5.4.25 Power Port I pull-down control register (PWR\_PDCRI)

Address offset: 0x64.

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. a standard APB access (3 for a write and 2 for a read).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 11:0 **PDy**: Port I pull-down bit y (y=0..11)

When set, this bit activates the pull-down on PI[y] when APC bit is set in PWR\_CR3 register.

### 5.4.26 PWR control register (PWR\_CR5)

Address offset: 0x80.

Reset value: 0x0000 0100. This register is not reset after a wakeup from Standby mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	R1MODE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								

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

Bit 8 **R1MODE**: Main regulator Range 1 mode

This bit is only valid for the main regulator in Range 1 and has no effect on Range 2. It is recommended to reset this bit when the system frequency is greater than 80 MHz. Refer to [Table 21: Range 1 boost mode configuration](#).

0: Main regulator in Range 1 boost mode.

1: Main regulator in Range 1 normal mode.

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

5.4.27 PWR register map and reset value table

Table 33. PWR register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x000	PWR_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VOS [1:0]	DBP	Res.	Res.	Res.	Res.	RRSTP	Res.	LPMS [2:0]	Res.		
	Reset value																			0	0	0	0	1	0				0		0	0	0	
0x004	PWR_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USV	IOSV	Res.	PVME4	PVME3	PVME2	PVME1	PLS [2:0]	Res.	Res.	Res.	
	Reset value																						0	0	0	0	0	0	0	0	0	0	0	0
0x008	PWR_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APC	Res.	RRS	Res.	Res.	Res.	Res.	EWUP5	EWUP4	EWUP3	EWUP2	EWUP1
	Reset value																		1			1		0	0	0	0	0	0	0	0	0	0	0
0x00C	PWR_CR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBR5	VBE	Res.	Res.	Res.	Res.	WP5	WP4	WP3	WP2	WP1
	Reset value																							0	0	0	0	0	0	0	0	0	0	0
0x010	PWR_SR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SBF	Res.	Res.	Res.	Res.	WUF5	WUF4	WUF3	WUF2	WUF1
	Reset value																									0	0	0	0	0	0	0	0	0
0x014	PWR_SR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	PWR_SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																	
0x020	PWR_PUCRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	PWR_PDCRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	PWR_PUCRB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	PWR_PDCRB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	PWR_PUCRC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	PWR_PDCRC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	PWR_PUCRD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	PWR_PDCRD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Table 33. 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	
0x040	PWR_PUCRE	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	PWR_PDCRE	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	PWR_PUCRF	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	PWR_PDCRF	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	PWR_PUCRG	Res	Res	Res	Res	Res	Res	Res	Res	Res	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
0x054	PWR_PDCRG	Res	Res	Res	Res	Res	Res	Res	Res	Res	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
0x058	PWR_PUCRH	Res	Res	Res	Res	Res	Res	Res	Res	Res	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
0x05C	PWR_PDCRH	Res	Res	Res	Res	Res	Res	Res	Res	Res	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
0x060	PWR_PUCRI	Res	Res	Res	Res	Res	Res	Res	Res	Res	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																																	
0x064	PWR_PDCRI	Res	Res	Res	Res	Res	Res	Res	Res	Res	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																																	
0x080	PWR_CR5	Res	Res	Res	Res	Res	Res	Res	Res	Res	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 2.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

## 6 Reset and clock control (RCC)

### 6.1 Reset

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

#### 6.1.1 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.

#### 6.1.2 System reset

A system reset sets all registers to their reset values except the reset flags in the clock control/status 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 firewall event (FIREWALL reset)
5. A software reset (SW reset) (see [Software reset](#))
6. Low-power mode security reset (see [Low-power mode security reset](#))
7. Option byte loader reset (see [Option byte loader reset](#))
8. A Brown-out reset

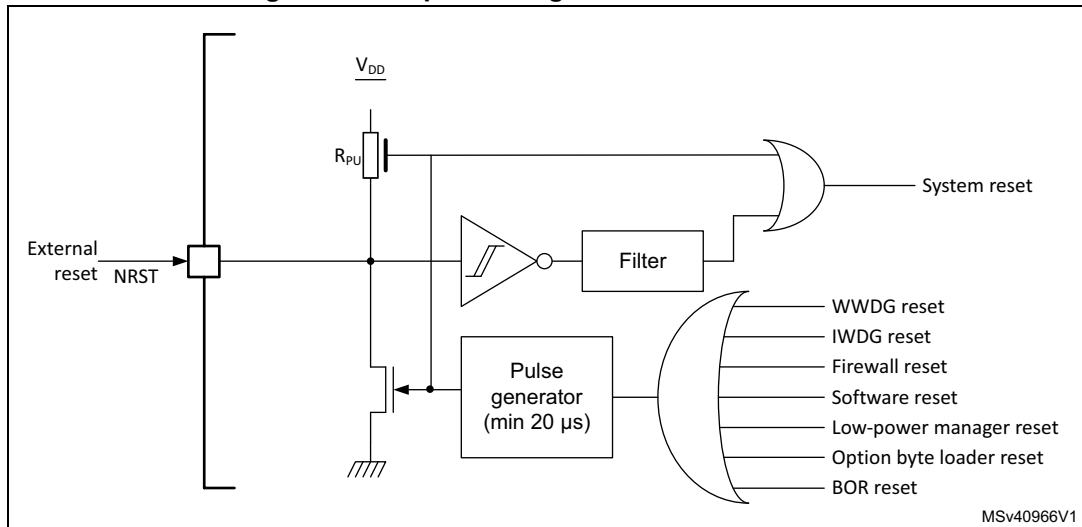
The reset source can be identified by checking the reset flags in the Control/Status register, RCC\_CSR (see [Section 6.4.30: Control/status register \(RCC\\_CSR\)](#)).

These sources act on the NRST pin and it is always kept low during the delay phase. The RESET service routine vector is fixed at address 0x0000\_0004 in the memory map.

The system reset signal provided to the device is output on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of 20  $\mu$ 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 on an internal reset, the internal pull-up  $R_{PU}$  is deactivated in order to save the power consumption through the pull-up resistor.

Figure 13. Simplified diagram of the reset circuit



### Software reset

The SYSRESETREQ bit in Cortex<sup>®</sup>-M4 Application Interrupt and Reset Control Register must be set to force a software reset on the device (refer to the *STM32F3*, *STM32F4*, *STM32L4* and *STM32L4+ Series Cortex<sup>®</sup>-M4* (PM0214)).

### Low-power mode security reset

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

1. Entering Standby mode: this type of reset is enabled by resetting nRST\_STDBY bit in User option Bytes. In this case, whenever a Standby mode entry sequence is successfully executed, the device is reset instead of entering Standby mode.
2. Entering Stop mode: this type of reset is enabled by resetting nRST\_STOP bit in User option bytes. In this case, whenever a Stop mode entry sequence is successfully executed, the device is reset instead of entering Stop mode.
3. Entering Shutdown mode: this type of reset is enabled by resetting nRST\_SHDW bit in User option bytes. In this case, whenever a Shutdown mode entry sequence is successfully executed, the device is reset instead of entering Shutdown mode.

For further information on the User Option Bytes, refer to [Section 3.4.1: Option bytes description](#).

### Option byte loader reset

The option byte loader reset is generated when the OBL\_LAUNCH bit (bit 27) is set in the FLASH\_CR register. This bit is used to launch the option byte loading by software.

## 6.1.3 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 *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.

## 6.2 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
- HSE oscillator clock, from 4 to 48 MHz
- 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:

- 32 kHz low speed internal RC (LSI RC) which drives the independent watchdog and optionally the RTC used for Auto-wakeup from Stop and Standby modes.
- 32.768 kHz low speed external crystal (LSE crystal) which optionally drives the real-time clock (RTCCLK).
- RC 48 MHz internal clock sources (HSI48) to potentially drive the USB FS, the SDMMC and the 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 frequency, the APB1 and APB2 domains. The maximum frequency of the AHB, the APB1 and the APB2 domains is 120 MHz.



All the peripheral clocks are derived from their bus clock (HCLK, PCLK1 or PCLK2) except:

- The 48 MHz clock, used for USB OTG FS, SDMMC and RNG. This clock is derived (selected by software) from one of the four following sources:
  - main PLL VCO (PLL48M1CLK)
  - PLLSAI1 VCO (PLL48M2CLK)
  - MSI clock
  - HSI48 internal oscillator

When the MSI clock is auto-trimmed with the LSE, it can be used by the USB OTG FS device.

When available, the HSI48 48 MHz clock can be coupled to the clock recovery system allowing adequate clock connection for the USB OTG FS (Crystal less solution).

- The ADCs clock which is derived (selected by software) from one of the following sources:
  - system clock (SYSCLK)
  - PLLSAI1 VCO (PLLADC1CLK)
- The U(S)ARTs clocks which are derived (selected by software) from one of the four following sources:
  - system clock (SYSCLK)
  - HSI16 clock
  - LSE clock
  - APB1 or APB2 clock (PCLK1 or PCLK2 depending on which APB is mapped the U(S)ART)

The wakeup from Stop mode is supported only when the clock is HSI16 or LSE.

- The I<sup>2</sup>Cs clocks which are derived (selected by software) from one of the three following sources:
  - system clock (SYSCLK)
  - HSI16 clock
  - APB1 clock (PCLK1)

The wakeup from Stop mode is supported only when the clock is HSI16.

- The SAI1 and SAI2 clocks which are derived (selected by software) from one of the following sources:
  - an external clock mapped on SAI1\_EXTCLK for SAI1 and SAI2\_EXTCLK for SAI2
  - PLLSAI1 VCO (PLLSAI1CLK)
  - PLLSAI2 VCO (PLLSAI2CLK)
  - main PLL VCO (PLLSAI3CLK)
  - HSI16 clock
- The DFSDM audio clock which is derived (selected by software) from one of the following sources:
  - SAI1 clock
  - HSI clock
  - MSI clock
- The LTDC clock. The LTDC clock is generated from a specific PLL (PLLSAI2).
- The DSI clock. To generate the DSI clocks, the high-speed external crystal (HSE) must be available.

- The DSI lanebyteclk clock: DSI Lane byte clock (high-speed clock divided by 8) which is output from the DSI-PHY or from a specific output of PLLSAI2 (frequency lower than 62.5 MHz) in the case when the DSI-PHY is off.
- The DSI host rxclkesc clock. The DSI RX escape mode clock is output from DSI-PHY (generated from DP0/DN0 even if the DSI-PHY is not clocked).
- The OctoSPI kernel clock which is derived (selected by software) from one of the following sources:
  - System clock,
  - PLL48M1CLK
  - MSI clock
- The low-power timers (LPTIMx) clock which are derived (selected by software) from one of the five following sources:
  - LSI clock
  - LSE clock
  - HSI16 clock
  - APB1 clock (PCLK1)
  - External clock mapped on LPTIMx\_IN1

The functionality in Stop mode (including wakeup) is supported only when the clock is LSI or LSE, or in external clock mode.

- The RTC clock which is derived (selected by software) from one of the three following sources:
  - LSE clock
  - LSI clock
  - HSE clock divided by 32

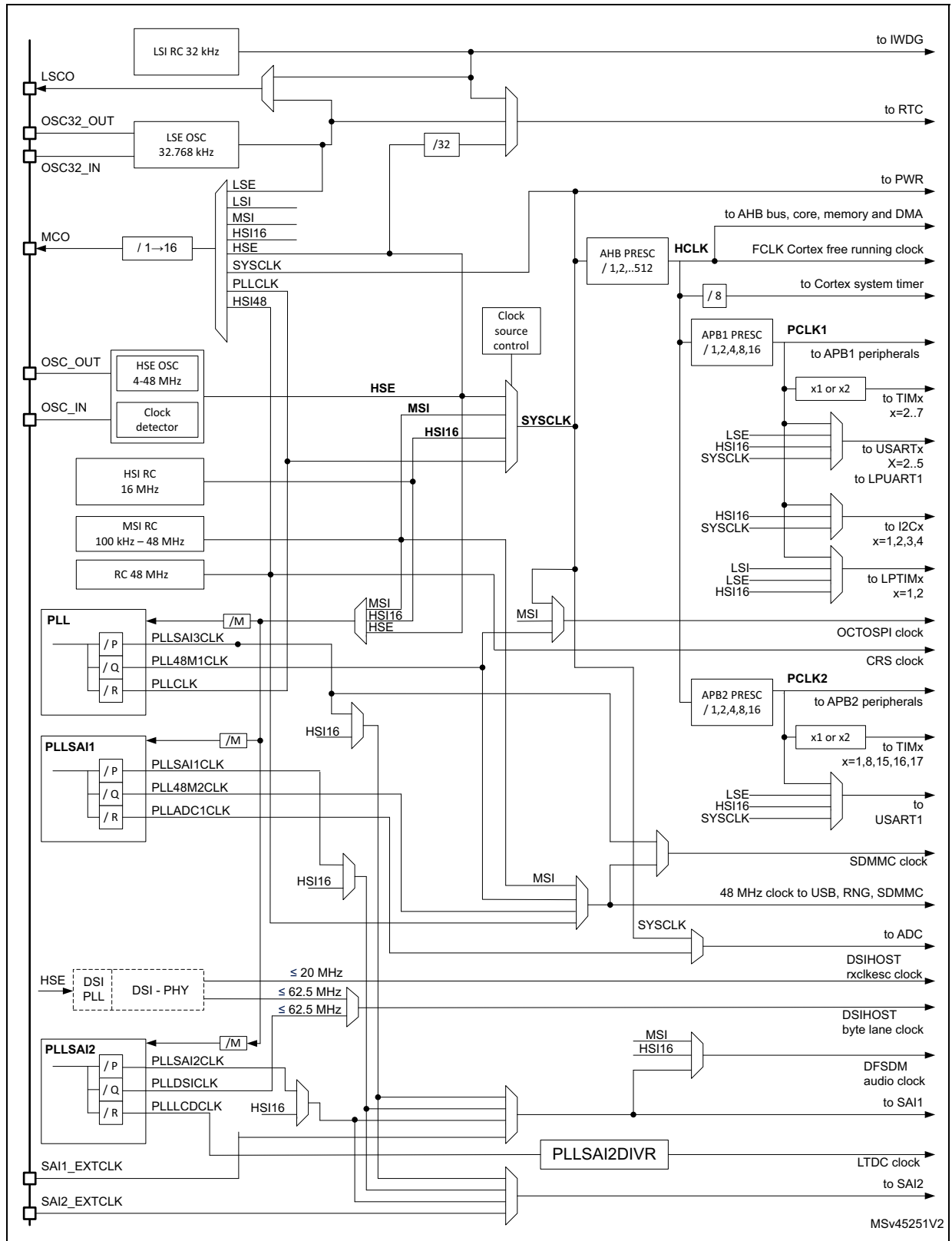
The functionality in Stop mode (including wakeup) is supported only when the clock is LSI or LSE.

- The IWDG clock which is always the LSI clock.

The RCC feeds the Cortex<sup>®</sup> System Timer (SysTick) external clock with the AHB clock (HCLK) divided by 8. The SysTick can work either with this clock or directly with the Cortex<sup>®</sup> clock (HCLK), configurable in the SysTick Control and Status Register.

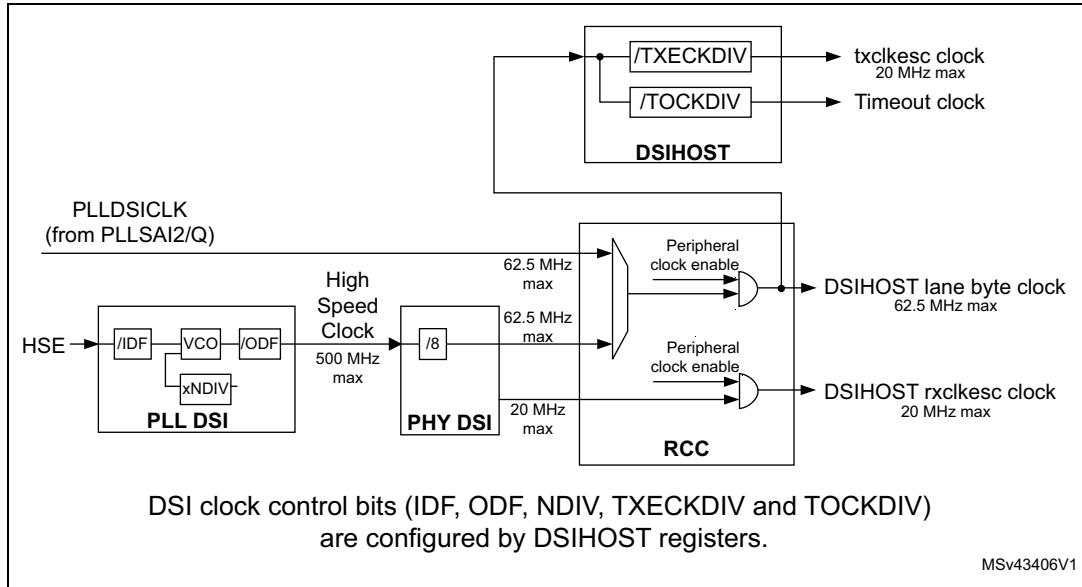
FCLK acts as Cortex<sup>®</sup>-M4 free-running clock. For more details refer to the *STM32F3*, *STM32F4*, *STM32L4* and *STM32L4+ Series Cortex<sup>®</sup>-M4* programming manual (PM0214).

Figure 14. Clock tree



1. For full details about the internal and external clock source characteristics, please refer to the “Electrical characteristics” section in your device datasheet.
2. The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). When the programmable factor is ‘1’, the AHB prescaler must be equal to ‘1’.

Figure 15. DSI clock tree



MSv43406V1

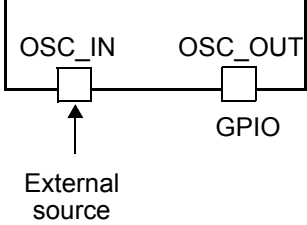
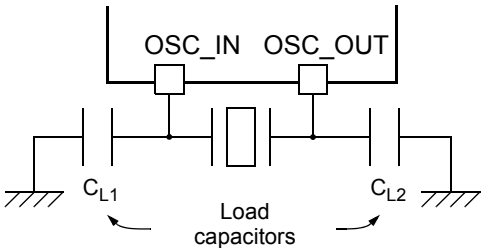
### 6.2.1 HSE clock

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

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

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

Figure 16. HSE/ LSE clock sources

Clock source	Hardware configuration
External clock	
Crystal/Ceramic resonators	

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

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

The associated hardware configuration is shown in [Figure 16](#). Refer to the electrical characteristics section of the *datasheet* for more details.

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

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

### External source (HSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 48 MHz. You select this mode by setting the HSEBYP and HSEON bits in the [Clock control register \(RCC\\_CR\)](#). The external clock signal (square, sinus or triangle) with ~40-60 % duty cycle depending on the frequency (refer to the *datasheet*) has to drive the OSC\_IN pin while the OSC\_OUT pin can be used a GPIO. See [Figure 16](#).

## 6.2.2 HSI16 clock

The HSI16 clock signal is generated from an internal 16 MHz RC Oscillator.

The HSI16 RC oscillator has the advantage of providing a clock source at low cost (no external components). It also has a faster startup time than the HSE crystal oscillator however, even 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 (Stop 0, Stop 1 or Stop 2). Refer to [Section 6.3: Low-power modes](#). It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to [Section 6.2.10: Clock security system \(CSS\)](#).

### Calibration

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

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

If the application is subject to voltage or temperature variations this may affect the RC oscillator speed. You can trim the HSI16 frequency in the application using the HSITRIM[6:0] in the [Internal clock sources calibration register \(RCC\\_ICSCR\)](#).

For more details on how to measure the HSI16 frequency variation, refer to [Section 6.2.17: Internal/external clock measurement with TIM15/TIM16/TIM17](#).

The HSIRDY flag in the [Clock control register \(RCC\\_CR\)](#) indicates if the HSI16 RC is stable or not. At startup, the HSI16 RC output clock is not released until this bit is set by hardware.

The HSI16 RC can be switched on and off using the HSION bit in the [Clock control register \(RCC\\_CR\)](#).

The HSI16 signal can also be used as a backup source (Auxiliary clock) if the HSE crystal oscillator fails. Refer to [Section 6.2.10: Clock security system \(CSS\) on page 228](#).

### 6.2.3 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 [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 Standby and Shutdown low-power modes. After restart from Reset, the MSI frequency is set to its default value 4 MHz. Refer to [Section 6.3: 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 6.3: Low-power modes](#). It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to [Section 6.2.10: Clock security system \(CSS\)](#).

The MSI RC oscillator has the advantage of providing a low-cost (no external components) low-power clock source. In addition, when used in PLL-mode with the LSE, it provides a very accurate clock source which can be used by the USB OTG FS device, and feed the main PLL to run the system at the maximum speed.

The MSIRDY flag in the [Clock control register \(RCC\\_CR\)](#) indicates whether 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 [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 [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 OTG FS device, saving the need of an external high-speed crystal.

#### 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 an ambient temperature, TA, of 25 °C. After reset, the factory calibration value is loaded in the MSICAL[7:0] bits in the [Internal clock sources calibration register \(RCC\\_ICSCR\)](#). If the application is subject to voltage or temperature variations, this may affect the RC oscillator speed. You can trim the MSI frequency in the application by using the MSITRIM[7:0] bits in the RCC\_ICSCR register. For more details on how to measure the MSI frequency variation please refer to [Section 6.2.17: Internal/external clock measurement with TIM15/TIM16/TIM17](#).

### 6.2.4 HSI48 clock

The HSI48 clock signal is generated from an internal 48 MHz RC oscillator and can be used directly for USB and for random number generator (RNG) as well as SDMMC.

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 please refer to [Section 7: Clock recovery system \(CRS\)](#).

The HSI48RDY flag in the Clock recovery RC 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 RC register (RCC\_CRRCR).

## 6.2.5 PLL

The device embeds 3 PLLs: PLL, PLLSAI1, PLLSAI2. Each PLL provides up to three independent outputs. The internal PLLs can be used to multiply the HSI16, HSE or MSI output clock frequency. The PLLs input frequency must be between 4 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 14: Clock tree](#) and [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 PLLON to 0 in [Clock control register \(RCC\\_CR\)](#).
2. Wait until PLLRDY is cleared. The PLL is now fully stopped.
3. Change the desired parameter.
4. Enable the PLL again by setting PLLON to 1.
5. Enable the desired PLL outputs by configuring PLLPEN, PLLQEN, PLLREN in [PLL configuration register \(RCC\\_PLLCFGR\)](#).

An interrupt can be generated when the PLL is ready, if enabled in the [Clock interrupt enable register \(RCC\\_CIER\)](#).

The same procedure is applied for changing the configuration of the PLLSAI1 or PLLSAI2:

1. Disable the PLLSAI1/PLLSAI2 by setting PLLSAI1ON/PLLSAI2ON to 0 in [Clock control register \(RCC\\_CR\)](#).
2. Wait until PLLSAI1RDY/PLLSAI2RDY is cleared. The PLLSAI1/PLLSAI2 is now fully stopped.
3. Change the desired parameter.
4. Enable the PLLSAI1/PLLSAI2 again by setting PLLSAI1ON/PLLSAI2ON to 1.
5. Enable the desired PLL outputs by configuring PLLSAI1PEN/PLLSAI2PEN, PLLSAI1QEN/PLLSAI2QEN, PLLSAI1REN/PLLSAI2REN in [PLLSAI1 configuration register \(RCC\\_PLLSAI1CFGR\)](#) and [PLLSAI2 configuration register \(RCC\\_PLLSAI2CFGR\)](#).

The PLL output frequency must not exceed 120 MHz.

The enable bit of each PLL output clock (PLLPEN, PLLQEN, PLLREN, PLLSAI1PEN, PLLSAI1QEN, PLLSAI1REN, PLLSAI2PEN and PLLSAI2REN) can be modified at any time



without stopping the corresponding PLL. PLLREN cannot be cleared if PLLCLK is used as system clock.

### 6.2.6 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 LSE crystal is switched on and off using the LSEON bit in *Backup domain control register (RCC\_BDCR)*. The crystal oscillator driving strength can be changed at runtime using the LSEDRV[1:0] bits in the *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 can not be increased if LSEON=1.

The LSERDY flag in the *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 *Clock interrupt enable register (RCC\_CIER)*.

#### External source (LSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 1 MHz. You select this mode by setting the LSEBYP and LSEON bits in the *AHB1 peripheral clocks enable in Sleep and Stop modes register (RCC\_AHB1SMENR)*. 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 16*.

### 6.2.7 LSI clock

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

The LSI RC can be switched on and off using the LSION bit in the *Control/status register (RCC\_CSR)*.

The LSIRDY flag in the *Control/status register (RCC\_CSR)* indicates if the LSI oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *Clock interrupt enable register (RCC\_CIER)*.

### 6.2.8 System clock (SYSCLK) selection

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

- MSI oscillator
- HSI16 oscillator
- HSE oscillator
- PLL

The system clock maximum frequency is 120 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 [Internal clock sources calibration register \(RCC\\_ICSCR\)](#) indicate which clock(s) is (are) ready and which clock is currently used as a system clock.

To switch from low speed to high speed or from high speed to low speed system clock, it is recommended to use a transition state with medium speed clock, for at least 1us.

Clock source switching conditions:

- Switching from HSE or HSI or MSI to PLL with AHB frequency (HCLK) higher than 80 MHz
- Switching from PLL with HCLK higher than 80 MHz to HSE or HSI or MSI

Transition state:

- Set the AHB prescaler HPRE[3:0] bits to divide the system frequency by 2
- Switch system clock to PLL
- Wait for at least 1us and then reconfigure AHB prescaler bits to the needed HCLK frequency

### 6.2.9 Clock source frequency versus voltage scaling

The following table gives the different clock source frequencies depending on the product voltage range.

**Table 34. Clock source frequency**

Product voltage range	Clock frequency			
	MSI	HSI16	HSE	PLL/PLLSAI1/PLLSAI2
Range 1 Boost mode	48 MHz	16 MHz	48 MHz	120 MHz
Range 1 Normal mode	48 MHz	16 MHz	48 MHz	80 MHz
Range 2	24 MHz range	16 MHz	26 MHz	26 MHz

### 6.2.10 Clock security system (CSS)

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

If a failure is detected on the HSE clock, the HSE oscillator is automatically disabled, a clock failure event is sent to the break input of the advanced-control timers (TIM1/TIM8 and TIM15/16/17) and an interrupt is generated to inform the software about the failure (Clock Security System Interrupt CSSI), allowing the MCU to perform rescue operations. The CSSI is linked to the Cortex®-M4 NMI (Non-Maskable Interrupt) exception vector.

*Note:* Once the CSS is enabled and if the HSE clock fails, the CSS interrupt occurs and a NMI is automatically generated. The NMI will be executed indefinitely unless the CSS interrupt pending bit is cleared. As a consequence, in the NMI ISR user must clear the CSS interrupt by setting the CSSC bit in the [Clock interrupt 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 [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.

### 6.2.11 Clock security system on LSE

A Clock Security System on LSE can be activated by software writing the LSECSSON bit in the [Backup domain control register \(RCC\\_BDCR\)](#). 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 LSION 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 [Clock interrupt enable register \(RCC\\_CIER\)](#), [Clock interrupt flag register \(RCC\\_CIFR\)](#), [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.

The frequency of LSE oscillator have to be higher than 30 kHz to avoid false positive CSS detection.

### 6.2.12 ADC clock

The ADC clock is derived from the system clock, or from the PLLSAI1 output. It can reach 120 MHz and can be divided by the following prescalers values:

1,2,4,6,8,10,12,16,32,64,128 or 256 by configuring the ADC1\_CCR register. It is asynchronous to the AHB clock. Alternatively, the ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). This programmable factor is configured using the CKMODE bit fields in the ADC123\_CCR.

If the programmed factor is '1', the AHB prescaler must be set to '1'.

### 6.2.13 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 [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 then 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.

### 6.2.14 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.

### 6.2.15 Watchdog clock

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

### 6.2.16 Clock-out capability

- MCO

The microcontroller clock output (MCO) capability allows the clock to be output onto the external MCO pin. One of eight clock signals can be selected as the MCO clock.

- LSI
- LSE
- SYSCLK
- HSI16
- HSI48
- HSE
- PLLCLK
- MSI

The selection is controlled by the MCOSEL[3:0] bits of the [Clock configuration register \(RCC\\_CFGR\)](#). The selected clock can be divided with the MCOPRE[2:0] field of the [Clock configuration register \(RCC\\_CFGR\)](#).

- LSCO

Another output (LSCO) allows a low speed clock to be output onto the external LSCO pin:

- LSI
- LSE

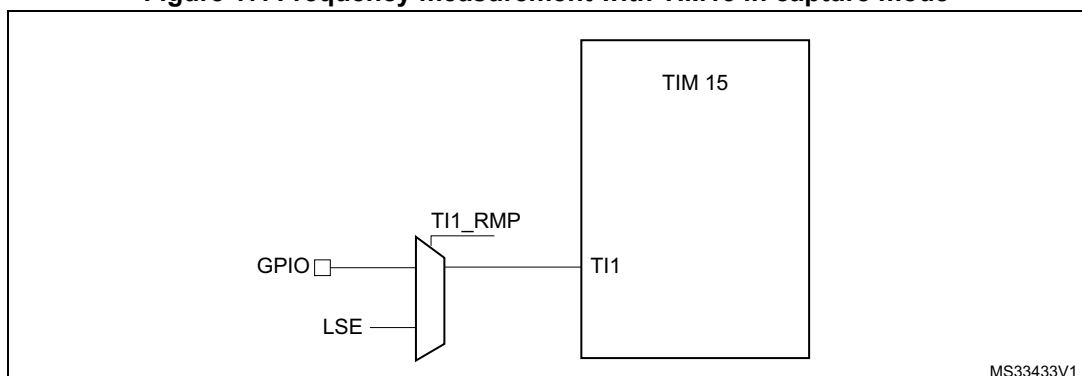
This output remains available in Stop (Stop 0, Stop 1 and Stop 2) and Standby modes. The selection is controlled by the LSCOSEL, and enabled with the LSCOEN in the [Backup domain control register \(RCC\\_BDCR\)](#).

The MCO clock output requires the corresponding alternate function selected on the MCO pin, the LSCO pin should be left in default POR state.

### 6.2.17 Internal/external clock measurement with TIM15/TIM16/TIM17

It is possible to indirectly measure the frequency of all on-board clock sources by mean of the TIM15, TIM16 or TIM17 channel 1 input capture, as represented on [Figure 17](#), [Figure 18](#) and [Figure 19](#).

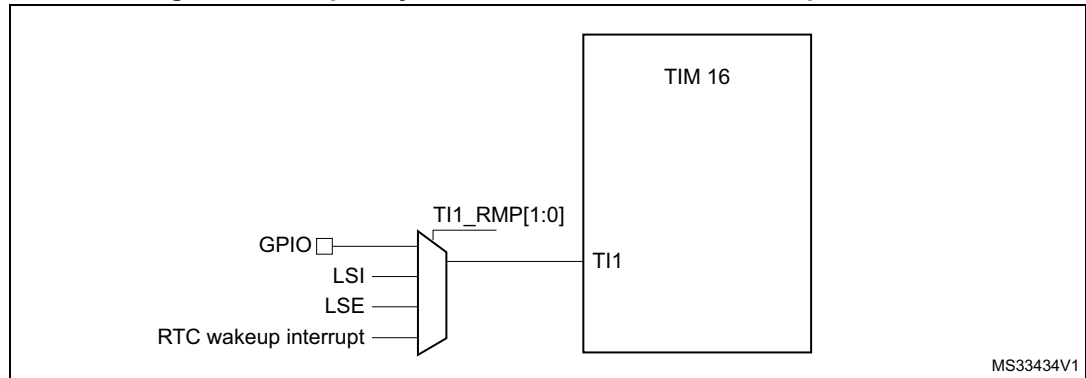
**Figure 17. Frequency measurement with TIM15 in capture mode**



The input capture channel of the Timer 15 can be a GPIO line or an internal clock of the MCU. This selection is performed through the TI1\_RMP bit in the TIM15\_OR register. The possibilities are the following ones:

- TIM15 Channel1 is connected to the GPIO. Refer to the alternate function mapping in the device datasheets.
- TIM15 Channel1 is connected to the LSE.

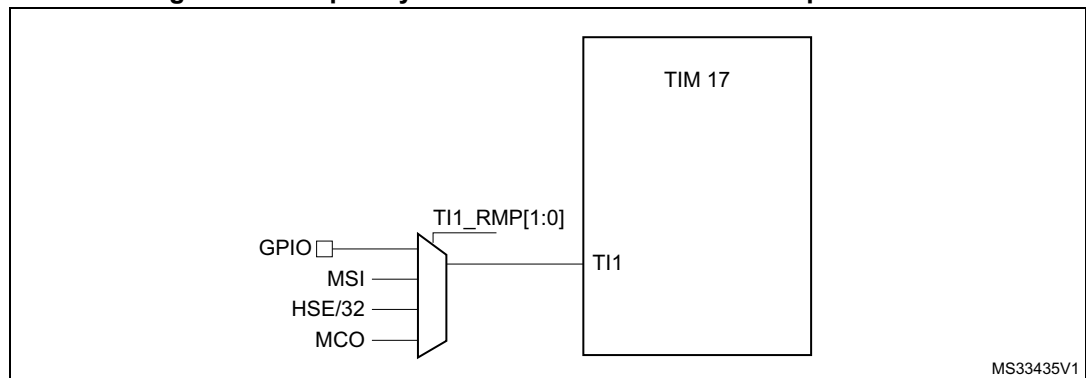
**Figure 18. Frequency measurement with TIM16 in capture mode**



The input capture channel of the Timer 16 can be a GPIO line or an internal clock of the MCU. This selection is performed through the 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 datasheets.
- 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\_CFGR).

### Calibration of the HSI16 and the MSI

For TIM15 and 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 tens of ppm's), it is possible to determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing, process, temperature and/or voltage related frequency deviations.

The MSI and HSI16 oscillator both have dedicated user-accessible calibration bits for this purpose.

The basic concept consists in providing a relative measurement (e.g. the HSI16/LSE ratio): the precision is therefore closely related to the ratio between the two clock sources. The higher the ratio is, the better the measurement will be.

If LSE is not available, HSE/32 will be the better option in order to reach the most precise calibration possible.

It is however not possible 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's input capture prescaler (up to 1 capture every 8 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 HSI16, but changing the reference clock. It will be necessary to connect LSI clock to the channel 1 input capture of the TIM16. Then define the HSE as system clock source, the number of his clock counts between consecutive edges of the LSI signal provides a measure of the internal low speed clock period.

The basic concept consists in providing a relative measurement (e.g. the HSE/LSI ratio): the precision is therefore closely related to the ratio between the two clock sources. The higher the ratio is, the better the measurement will be.

### 6.2.18 Peripheral clock enable register (RCC\_AHBxENR, RCC\_APBxENRy)

Each peripheral clock can be enabled by the xxxxEN bit of the RCC\_AHBxENR, RCC\_APBxENRy registers.

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

The enable bit has a synchronization mechanism to create a glitch free clock for the peripheral. After the enable bit is set, there is a 2 clock cycles delay before the clock be active.

**Caution:** Just after enabling the clock for a peripheral, software must wait for a delay before accessing the peripheral registers.

## 6.3 Low-power modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep and Low Power Sleep modes stops the CPU clock. The memory interface clocks (Flash, SRAM1, SRAM2 and SRAM3 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 all the clocks in the  $V_{CORE}$  domain and disables the three PLL, the HSI16, the MSI and the HSE oscillators.

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 and LPUARTs 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).

- Standby and Shutdown modes stops all the clocks in the  $V_{CORE}$  domain and disables the PLL, the HSI16, the MSI and the HSE oscillators.

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

When leaving the Stop modes (Stop 0, Stop 1 or Stop 2), the system clock is either MSI or HSI16, depending on the software configuration of the STOPWUCK bit in the RCC\_CFGR register. 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.

When leaving the Standby and Shutdown modes, the system clock is MSI. The MSI frequency at wakeup from Standby mode is configured with the MSISRANGE is the RCC\_CSR register, from 1 to 8 MHz. 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.



## 6.4 RCC registers

### 6.4.1 Clock control register (RCC\_CR)

Address offset: 0x00

Reset value: 0x0000 0063. HSEBYP is not affected by 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.	PLLSAI2RDY	PLLSAI2ON	PLLSAI1RDY	PLLSAI1ON	PLLRDY	PLLON	Res.	Res.	Res.	Res.	CSSON	HSEBYP	HSERDY	HSEON
		r	rw	r	rw	r	rw					rs	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIAFS	HSIRDY	HSIKERON	HSION	MSIRANGE[3:0]				MSIRGSEL	MSIPLLEN	MSIRDY	MSION
				rw	r	rw	rw	rw	rw	rw	rw	rs	rw	r	rw

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

Bit 29 **PLLSAI2RDY**: SAI2 PLL clock ready flag

Set by hardware to indicate that the PLLSAI2 is locked.

0: PLLSAI2 unlocked

1: PLLSAI2 locked

Bit 28 **PLLSAI2ON**: SAI2 PLL enable

Set and cleared by software to enable PLLSAI2.

Cleared by hardware when entering Stop, Standby or Shutdown mode.

0: PLLSAI2 OFF

1: PLLSAI2 ON

Bit 27 **PLLSAI1RDY**: SAI1 PLL clock ready flag

Set by hardware to indicate that the PLLSAI1 is locked.

0: PLLSAI1 unlocked

1: PLLSAI1 locked

Bit 26 **PLLSAI1ON**: SAI1 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**: Main PLL clock ready flag

Set by hardware to indicate that the main PLL is locked.

0: PLL unlocked

1: PLL locked

Bit 24 **PLLON**: Main PLL enable

Set and cleared by software to enable the main 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 23:20 Reserved, must be kept at reset value.

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

Set by software to enable the clock security system. When CSSON is set, the clock detector is enabled by hardware when the HSE oscillator is ready, and disabled by hardware if a HSE clock failure is detected. This bit is set only and is cleared by reset.

0: Clock security system OFF (clock detector OFF)

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

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

Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit set, to be used by the device. The HSEBYP bit can be written only if the HSE oscillator is disabled.

0: HSE crystal oscillator not bypassed

1: HSE crystal oscillator bypassed with external clock

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

Set by hardware to indicate that the HSE oscillator is stable.

0: HSE oscillator not ready

1: HSE oscillator ready

*Note: Once the HSEON bit is cleared, HSERDY goes low after 6 HSE clock cycles.*

Bit 16 **HSEON**: HSE clock enable

Set and cleared by software.

Cleared by hardware to stop the HSE oscillator when entering Stop, Standby or Shutdown mode. This bit cannot be reset if the HSE oscillator is used directly or indirectly as the system clock.

0: HSE oscillator OFF

1: HSE oscillator ON

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

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 in 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

Set by hardware to indicate that HSI16 oscillator is stable. This bit is set only when HSI16 is enabled by software by setting HSION.

0: HSI16 oscillator not ready

1: HSI16 oscillator ready

*Note: Once the HSION bit is cleared, HSIRDY goes low after 6 HSI16 clock cycles.*

Bit 9 **HSIKERON**: HSI16 always enable for peripheral kernels.

Set and cleared by software to force HSI16 ON even in Stop modes. The HSI16 can only feed USARTs and I<sup>2</sup>Cs peripherals configured with HSI16 as kernel clock. Keeping the HSI16 ON in Stop mode allows to avoid 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 MSIRGSEL is set. 12 frequency ranges are available:

0000: range 0 around 100 kHz

0001: range 1 around 200 kHz

0010: range 2 around 400 kHz

0011: range 3 around 800 kHz

0100: range 4 around 1M Hz

0101: range 5 around 2 MHz

0110: range 6 around 4 MHz (reset value)

0111: range 7 around 8 MHz

1000: range 8 around 16 MHz

1001: range 9 around 24 MHz

1010: range 10 around 32 MHz

1011: range 11 around 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 **MSIRGSEL**: MSI clock range selection

Set by software to select the MSI clock range with MSIRANGE[3:0]. Write 0 has no effect.

After a standby or a reset MSIRGSEL is at 0 and the MSI range value is provided by MSIRANGE in CSR register.

0: MSI Range is provided by MSIRANGE[3:0] in RCC\_CSR register

1: MSI Range is provided by MSIRANGE[3:0] in the RCC\_CR register

Bit 2 **MSIPLLEN**: MSI clock PLL enable

Set and cleared by software to enable/ disable the PLL part of the MSI clock source.

MSIPLLEN must be enabled after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware). There is a hardware protection to avoid enabling MSIPLLEN 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

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

### 6.4.2 Internal clock sources calibration register (RCC\_ICSCR)

Address offset: 0x04

Reset value: 0x40XX 00XX 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.	HSITRIM[6:0]							HSICAL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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 16, which, when added to the HSICAL value, should trim the HSI16 to 16 MHz ± 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.

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.

### 6.4.3 Clock configuration register (RCC\_CFGR)

Address offset: 0x08

Reset value: 0x0000 0000

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

1 or 2 wait states inserted only if the access occurs during 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.	Res.	Res.	Res.
	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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	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

0101: Main PLL clock selected

0110: LSI clock selected

0111: LSE clock selected

1000: Internal HSI48 clock selected

Others: Reserved

*Note: This clock output may have some truncated cycles at startup or during MCO clock source switching.*

Bits 23:16 Reserved, must be kept at reset value.

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 Clock Security System is enabled by HSECSSON in 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]**: APB high-speed prescaler (APB2)

Set and cleared by software to control the division factor of the APB2 clock (PCLK2).

0xx: HCLK not divided  
100: HCLK divided by 2  
101: HCLK divided by 4  
110: HCLK divided by 8  
111: HCLK divided by 16

Bits 10:8 **PPRE1[2:0]**: APB low-speed prescaler (APB1)

Set and cleared by software to control the division factor of the APB1 clock (PCLK1).

0xx: HCLK not divided  
100: HCLK divided by 2  
101: HCLK divided by 4  
110: HCLK divided by 8  
111: HCLK divided by 16

Bits 7:4 **HPRE[3:0]**: AHB prescaler

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

**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 please refer to [Section 5.1.7: Dynamic voltage scaling management](#)). After a write operation to these bits and before decreasing the voltage range, this register must be read to be sure that the new value has been taken into account.

0xxx: SYSCLK not divided  
1000: SYSCLK divided by 2  
1001: SYSCLK divided by 4  
1010: SYSCLK divided by 8  
1011: SYSCLK divided by 16  
1100: SYSCLK divided by 64  
1101: SYSCLK divided by 128  
1110: SYSCLK divided by 256  
1111: SYSCLK divided by 512

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 Standby or Shutdown 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

### 6.4.4 PLL configuration register (RCC\_PLLCFGR)

Address offset: 0x0C

Reset value: 0x0000 1000

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

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

- $f(\text{VCO clock}) = f(\text{PLL clock input}) \times (\text{PLLN} / \text{PLLM})$
- $f(\text{PLL\_P}) = f(\text{VCO clock}) / \text{PLL P}$
- $f(\text{PLL\_Q}) = f(\text{VCO clock}) / \text{PLL Q}$
- $f(\text{PLL\_R}) = f(\text{VCO clock}) / \text{PLL R}$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLLPDIV[4:0]				PLLR[1:0]		PLLREN	Res.	PLLQ[1:0]		PLLQEN	Res.	Res.	PLL P	PLL PEN	
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]						PLLM[3:0]				Res.	Res.	PLLSRC[1:0]		
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw

Bits 31:27 **PLLPDIV[4:0]**: Main PLLP division factor for PLLSAI3CLK

Set and cleared by software to control the SAI1 or SAI2 clock frequency. PLLSAI3CLK output clock frequency = VCO frequency / PLLPDIV.

00000: PLLSAI3CLK is controlled by the bit PLLP

00001: Reserved.

00010: PLLSAI3CLK = VCO / 2

....

11111: PLLSAI3CLK = VCO / 31

Bits 26:25 **PLLR[1:0]**: Main PLL division factor for PLLCLK (system clock)

Set and cleared by software to control the frequency of the main PLL output clock PLLCLK. This output can be selected as system clock. These bits can be written only if PLL is disabled.

PLLCLK output clock frequency = VCO frequency / PLLR with PLLR = 2, 4, 6, or 8

00: PLLR = 2

01: PLLR = 4

10: PLLR = 6

11: PLLR = 8

**Caution:** The software has to set these bits correctly not to exceed 120 MHz on this domain.

Bit 24 **PLLREN**: Main PLL PLLCLK output enable

Set and reset by software to enable the PLLCLK output of the main PLL (used as system clock).

This bit cannot be written when PLLCLK output of the PLL is used as System Clock.

In order to save power, when the PLLCLK output of the PLL is not used, the value of PLLREN should be 0.

0: PLLCLK output disable

1: PLLCLK output enable

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **PLLQ[1:0]**: Main PLL division factor for PLL48M1CLK (48 MHz clock).

Set and cleared by software to control the frequency of the main PLL output clock PLL48M1CLK. This output can be selected for USB, RNG, SDMMC (48 MHz clock). These bits can be written only if PLL is disabled.

PLL48M1CLK output clock frequency = VCO frequency / PLLQ with PLLQ = 2, 4, 6, or 8

00: PLLQ = 2

01: PLLQ = 4

10: PLLQ = 6

11: PLLQ = 8

**Caution:** The software has to set these bits correctly not to exceed 120 MHz on this domain.

Bit 20 **PLLQEN**: Main PLL PLL48M1CLK output enable

Set and reset by software to enable the PLL48M1CLK output of the main PLL.

In order to save power, when the PLL48M1CLK output of the PLL is not used, the value of PLLQEN should be 0.

0: PLL48M1CLK output disable

1: PLL48M1CLK output enable

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **PLL P**: Main PLL division factor for PLLSAI3CLK (SAI1 and SAI2 clock).

Set and cleared by software to control the frequency of the main PLL output clock PLLSAI3CLK. This output can be selected for SAI1 or SAI2. These bits can be written only if PLL is disabled.

When the PLLPDIV[4:0] is set to "00000" PLLSAI3CLK output clock frequency = VCO frequency / PLLP with PLLP = 7, or 17

0: PLLP = 7

1: PLLP = 17

**Caution:** The software has to set these bits correctly not to exceed 120 MHz on this domain.

Bit 16 **PLL PEN**: Main PLL PLLSAI3CLK output enable

Set and reset by software to enable the PLLSAI3CLK output of the main PLL.

In order to save power, when the PLLSAI3CLK output of the PLL is not used, the value of PLLPEN should be 0.

0: PLLSAI3CLK output disable

1: PLLSAI3CLK 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 x PLLN with  $8 \leq \text{PLLN} \leq 127$

0000000: PLLN = 0 wrong configuration

0000001: PLLN = 1 wrong configuration

...

0000111: PLLN = 7 wrong configuration

0001000: PLLN = 8

0001001: PLLN = 9

...

1111111: PLLN = 127

**Caution:** The software has to set correctly these bits to assure that the VCO output frequency is between 64 and 344 MHz.

Bits 7:4 **PLLM**: Division factor for the main PLLinput clock

Set and cleared by software to divide the PLL 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  $1 \leq \text{PLLM} \leq 16$

0000: PLLM = 1

0001: PLLM = 2

0010: PLLM = 3

0011: PLLM = 4

0100: PLLM = 5

0101: PLLM = 6

0110: PLLM = 7

0111: PLLM = 8

**Caution:** The software has to set these bits correctly to ensure that the VCO input frequency ranges from 2.66 MHz to 8 MHz.

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

Bits 1:0 **PLLSRC**: Main PLL entry clock source

Set and cleared by software to select PLL clock source. These bits can be written only when PLL disabled.

In order to save power, when no PLL is used, the value of PLLSRC should be 00.

00: No clock sent to PLL

01: MSI clock selected as PLL clock entry

10: HSI16 clock selected as PLL clock entry

11: HSE clock selected as PLL clock entry

### 6.4.5 PLLSAI1 configuration register (RCC\_PLLSAI1CFGR)

Address offset: 0x10

Reset value: 0x0000 1000

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

This register is used to configure the PLLSAI1 clock outputs according to the formulas:

- $f(\text{VCOSAI1 clock}) = f(\text{PLL clock input}) \times (\text{PLLSAI1N} / \text{PLLSAI1M})$
- $f(\text{PLLSAI1\_P}) = f(\text{VCOSAI1 clock}) / \text{PLLSAI1PDIV}$
- $f(\text{PLLSAI1\_Q}) = f(\text{VCOSAI1 clock}) / \text{PLLSAI1Q}$

- $f(\text{PLLSAI1\_R}) = f(\text{VCOSAI1 clock}) / \text{PLLSAI1R}$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLLSAI1PDIV[4:0]					PLLSAI1R[1:0]		PLLSAI1REN	Res.	PLLSAI1Q[1:0]		PLLSAI1QEN	Res.	Res.	PLLSAI1P	PLLSAI1PEN
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.	PLLSAI1N[6:0]						PLLSAI1M[3:0]				Res.	Res.	Res.	Res.	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:27 **PLLSAI1PDIV[4:0]**: PLLSAI1 division factor for PLLSAI1CLK  
 Set and cleared by software to control the SAI1 or SAI2 clock frequency. PLLSAI1CLK output clock frequency = VCOSAI1 frequency / PLLSAI1PDIV.  
 00000: PLLSAI1CLK is controlled by the bit PLLSAI1P  
 00001: Reserved.  
 00010: PLLSAI1CLK = VCOSAI1 / 2  
 ....  
 11111: PLLSAI1CLK = VCOSAI1 / 31

*Note:* This bit can be written only when the PLLSAI1 is disabled.

Bits 26:25 **PLLSAI1R[1:0]**: PLLSAI1 division factor for PLLADC1CLK (ADC clock)  
 Set and cleared by software to control the frequency of the PLLSAI1 output clock PLLADC1CLK. This output can be selected as ADC clock. These bits can be written only if PLLSAI1 is disabled.  
 PLLADC1CLK output clock frequency = VCOSAI1 frequency / PLLSAI1R with PLLSAI1R = 2, 4, 6, or 8  
 00: PLLSAI1R = 2  
 01: PLLSAI1R = 4  
 10: PLLSAI1R = 6  
 11: PLLSAI1R = 8

Bit 24 **PLLSAI1REN**: PLLSAI1 PLLADC1CLK output enable  
 Set and reset by software to enable the PLLADC1CLK output of the PLLSAI1 (used as clock for ADC).  
 In order to save power, when the PLLADC1CLK output of the PLLSAI1 is not used, the value of PLLSAI1REN should be 0.  
 0: PLLADC1CLK output disable  
 1: PLLADC1CLK output enable

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **PLLSAI1Q[1:0]**: PLLSAI1 division factor for PLL48M2CLK (48 MHz clock)  
 Set and cleared by software to control the frequency of the PLLSAI1 output clock PLL48M2CLK. This output can be selected for USB, RNG, SDMMC (48 MHz clock). These bits can be written only if PLLSAI1 is disabled.  
 PLL48M2CLK output clock frequency = VCOSAI1 frequency / PLLSAI1Q with PLLSAI1Q = 2, 4, 6, or 8  
 00: PLLSAI1Q = 2  
 01: PLLSAI1Q = 4  
 10: PLLSAI1Q = 6  
 11: PLLSAI1Q = 8

**Caution:** The software has to set these bits correctly not to exceed 120 MHz on this domain.

Bit 20 **PLLSAI1QEN**: PLLSAI1 PLL48M2CLK output enable

Set and reset by software to enable the PLL48M2CLK output of the PLLSAI1.

In order to save power, when the PLL48M2CLK output of the PLLSAI1 is not used, the value of PLLSAI1QEN should be 0.

0: PLL48M2CLK output disable

1: PLL48M2CLK output enable

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **PLLSAI1P**: PLLSAI1 division factor for PLLSAI1CLK (SAI1 or SAI2 clock).

Set and cleared by software to control the frequency of the PLLSAI1 output clock PLLSAI1CLK. This output can be selected for SAI1 or SAI2. These bits can be written only if PLLSAI1 is disabled.

When the PLLSAI1PDIV[4:0] is set to "00000", PLLSAI1CLK output clock frequency = VCOSAI1 frequency / PLLSAI1P with PLLSAI1P = 7, or 17

0: PLLSAI1P = 7

1: PLLSAI1P = 17

Bit 16 **PLLSAI1PEN**: PLLSAI1 PLLSAI1CLK output enable

Set and reset by software to enable the PLLSAI1CLK output of the PLLSAI1.

In order to save power, when the PLLSAI1CLK output of the PLLSAI1 is not used, the value of PLLSAI1PEN should be 0.

0: PLLSAI1CLK output disable

1: PLLSAI1CLK output enable

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **PLLSAI1N[6:0]**: 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.

$VCOSAI1 \text{ output frequency} = VCOSAI1 \text{ input frequency} \times PLLSAI1N$

with  $8 \leq PLLSAI1N \leq 127$

0000000: PLLSAI1N = 0 wrong configuration

0000001: PLLSAI1N = 1 wrong configuration

...

0000111: PLLSAI1N = 7 wrong configuration

0001000: PLLSAI1N = 8

0001001: PLLSAI1N = 9

...

1111111: PLLSAI1N = 127

**Caution:** The software has to set correctly these bits to ensure that the VCO output frequency is between 64 and 344 MHz.

Bits 7:4 **PLLSAI1M**: Division factor for PLLSAI1 input clock

Set and reset by software to divide the PLLSAI1 input clock before the VCO.

These bits can be written only when PLLSAI1 is disabled.

$VCO \text{ input frequency} = PLLSAI1 \text{ input clock frequency} / PLLSAI1M$  with  $1 \leq PLLSAI1M \leq 16$

0000: PLLSAI1M = 1

0001: PLLSAI1M = 2

0010: PLLSAI1M = 3

0011: PLLSAI1M = 4

0100: PLLSAI1M = 5

0101: PLLSAI1M = 6

0110: PLLSAI1M = 7

0111: PLLSAI1M = 8

1000: PLLSAI1M = 9

...

1111: PLLSAI1M = 16

**Caution:** The software has to set these bits correctly to ensure that the VCO input frequency ranges from 2.66 to 8 MHz.

Bits 3:0 Reserved, must be kept at reset value.

### 6.4.6 PLLSAI2 configuration register (RCC\_PLLSAI2CFGR)

Address offset: 0x14

Reset value: 0x0000 1000

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

This register is used to configure the PLLSAI2 clock outputs according to the formulas:

- $f(\text{VCOSAI2 clock}) = f(\text{PLL clock input}) \times (\text{PLLSAI2N} / \text{PLLSAI2M})$
- $f(\text{PLLSAI2\_P}) = f(\text{VCOSAI2 clock}) / \text{PLLSAI2PDIV}$
- $f(\text{PLLSAI2\_R}) = f(\text{VCOSAI2 clock}) / \text{PLLSAI2R}$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLLSAI2PDIV[4:0]				PLLSAI2R[1:0]		PLLSAI2REN	Res.	PLLSAI2Q			PLLSAI2QEN	Res.	Res.	PLLSAI2P	PLLSAI2PEN
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.	PLLSAI2N[6:0]						PLLSAI2M[3:0]				Res.	Res.	Res.	Res.	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:27 **PLLSAI2PDIV[4:0]**: PLLSAI2 division factor for PLLSAI2CLK  
 Set and cleared by software to control the SAI1 or SAI2 clock frequency. PLLSAI2CLK output clock  
 $\text{frequency} = \text{VCOSAI2 frequency} / \text{PLLSAI2PDIV}$ .  
 00000: PLLSAI2CLK is controlled by the bit PLLSAI2P  
 00001: Reserved.  
 00010: PLLSAI2CLK = VCOSAI2 / 2  
 ....  
 11111: PLLSAI2CLK = VCOSAI2 / 31

Bits 26:25 **PLLSAI2R[1:0]**: PLLSAI2 division factor for PLLLCDCLK (LTDC clock)  
 Set and cleared by software to control the frequency of the PLLSAI2 output clock PLLLCDCLK. This output can be selected as ADC clock. These bits can be written only if PLLSAI2 is disabled.  
 $\text{PLLLCDCLK output clock frequency} = \text{VCOSAI2 frequency} / \text{PLLSAI2R}$  with PLLSAI2R = 2, 4, 6, or 8  
 00: PLLSAI2R = 2  
 01: PLLSAI2R = 4  
 10: PLLSAI2R = 6  
 11: PLLSAI2R = 8

Bit 24 **PLLSAI2REN**: PLLSAI2 PLLLCDCLK output enable  
 Set and reset by software to enable the PLLLCDCLK output of the PLLSAI2 (used as clock for ADC).  
 In order to save power, when the PLLLCDCLK output of the PLLSAI2 is not used, the value of PLLSAI2REN should be 0.  
 0: PLLLCDCLK output disable  
 1: PLLLCDCLK output enable

Bit 23 Reserved, must be kept at reset value.

- Bits 22:21 **PLLSAI2Q[1:0]**: PLLSAI2 PLLDSICLK output enable.  
Set and cleared by software to control the frequency of the DSI clock.  
These bits can be written only if PLLSAI2 is disabled.  
PLLDSICLK output clock frequency = VCOSAI2 frequency / PLLSAI2Q with PLLSAI2Q = 2, 4, 6, or 8  
00: PLLSAI2Q = 2  
01: PLLSAI2Q = 4  
10: PLLSAI2Q = 6  
11: PLLSAI2Q = 8
- Bit 20 **PLLSAI2QEN**: PLLSAI2 division factor for PLLDSICLK (DSI clock).  
Set and reset by software to enable the PLLDSICLK (DSI clock) output of the PLLSAI2 (used as USB clock).  
In order to save power, when the PLLDSICLK output of the PLLSAI2 is not used, the value of PLLSAI2PEN should be 0.  
0: PLLDSICLK output disable  
1: PLLDSICLK output enable
- Bits 19:18 Reserved, must be kept at reset value.
- Bit 17 **PLLSAI2P**: PLLSAI2 division factor for PLLSAI2CLK (SAI1 or SAI2 clock).  
Set and cleared by software to control the frequency of the PLLSAI2 output clock PLLSAI2CLK. This output can be selected for SAI1 or SAI2. These bits can be written only if PLLSAI2 is disabled.  
(when the PLLSAI2PDIV[4:0] is set to "00000"), PLLSAI2CLK output clock frequency = VCOSAI2 frequency / PLLSAI2P with PLLSAI2P = 7, or 17  
0: PLLSAI2P = 7  
1: PLLSAI2P = 17
- Bit 16 **PLLSAI2PEN**: PLLSAI2 PLLSAI2CLK output enable  
Set and reset by software to enable the PLLSAI2CLK output of the PLLSAI2.  
In order to save power, when the PLLSAI2CLK output of the PLLSAI2 is not used, the value of PLLSAI2PEN should be 0.  
0: PLLSAI2CLK output disable  
1: PLLSAI2CLK output enable
- Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **PLLSAI2N[6:0]**: PLLSAI2 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 PLLSAI2 is disabled.

VCOSAI2 output frequency = VCOSAI2 input frequency x PLLSAI2N  
with  $8 \leq \text{PLLSAI2N} \leq 127$

0000000: PLLSAI2N = 0 wrong configuration

0000001: PLLSAI2N = 1 wrong configuration

...

0000111: PLLSAI2N = 7 wrong configuration

0001000: PLLSAI2N = 8

0001001: PLLSAI2N = 9

...

1111111: PLLSAI2N = 127

**Caution:** The software has to set correctly these bits to ensure that the VCO output frequency is between 64 and 344 MHz.

Bits 7:4 **PLLSAI2M**: Division factor for PLLSAI2 input clock

Set and reset by software to divide the PLLSAI2 input clock before the VCO.

These bits can be written only when PLLSAI2 is disabled.

VCO input frequency = PLLSAI2 input clock frequency / PLLM with  $1 \leq \text{PLLSAI2M} \leq 16$

0000: PLLSAI2M = 1

0001: PLLSAI2M = 2

0010: PLLSAI2M = 3

0011: PLLSAI2M = 4

0100: PLLSAI2M = 5

0101: PLLSAI2M = 6

0110: PLLSAI2M = 7

0111: PLLSAI2M = 8

1000: PLLSAI2M = 9

...

1111: PLLSAI2M = 16

**Caution:** The software has to set these bits correctly to ensure that the VCO input frequency ranges from 2.66 to 8 MHz.

Bits 3:0 Reserved, must be kept at reset value.

### 6.4.7 Clock interrupt enable register (RCC\_CIER)

Address offset: 0x18

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	HSI48 RDYIE	LSECS SIE	Res.	PLLSAI 2RDYI E	PLLSAI 1RDYI E	PLL RDYIE	HSE RDYIE	HSI RDYIE	MSI RDYIE	LSE RDYIE	LSI RDYIE
					rw	rw		rw	rw	rw	rw	rw	rw	rw	rw



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

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

Bit 8 Reserved, must be kept at reset value.

Bit 7 **PLLSAI2RDYIE**: PLLSAI2 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by PLLSAI2 lock.

0: PLLSAI2 lock interrupt disabled

1: PLLSAI2 lock interrupt enabled

Bit 6 **PLLSAI1RDYIE**: PLLSAI1 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by PLLSAI1L 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 **LSIRDYIE**: LSI ready interrupt enable  
 Set and cleared by software to enable/disable interrupt caused by the LSI oscillator stabilization.  
 0: LSI ready interrupt disabled  
 1: LSI ready interrupt enabled

### 6.4.8 Clock interrupt flag register (RCC\_CIFR)

Address offset: 0x1C

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.	Res.	HSI48 RDYF	LSECS SF	CSSF	PLLSAI 2RDYF	PLLSAI 1RDYF	PLL RDYF	HSE RDYF	HSI RDYF	MSI RDYF	LSE RDYF	LSI RDYF
					r	r	r	r	r	r	r	r	r	r	r

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

- 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 [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**: 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 **PLLSAI2RDYF**: PLLSAI2 ready interrupt flag  
Set by hardware when the PLLSAI2 locks and PLLSAI2RDYDIE is set.  
Cleared by software setting the PLLSAI2RDYC bit.  
0: No clock ready interrupt caused by PLLSAI2 lock  
1: Clock ready interrupt caused by PLLSAI2 lock
- 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 [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 **LSIRDYF**: LSI ready interrupt flag  
Set by hardware when the LSI clock becomes stable and LSIRDYDIE is set.  
Cleared by software setting the LSIRDYC bit.  
0: No clock ready interrupt caused by the LSI oscillator  
1: Clock ready interrupt caused by the LSI oscillator

#### 6.4.9 Clock interrupt clear register (RCC\_CICR)

Address offset: 0x20

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	HSI48 RDYC	LSECS SC	CSSC	PLLSAI 2RDYC	PLLSAI 1RDYC	PLL RDYC	HSER DYC	HSIRD YC	MSIRD YC	LSERD YC	LSIRDY C
						w	w	w	w	w	w	w	w	w	w

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

- 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**: Clock security system interrupt clear  
 This bit is set by software to clear the CSSF flag.  
 0: No effect  
 1: Clear CSSF flag
- Bit 7 **PLLSAI2RDYC**: PLLSAI2 ready interrupt clear  
 This bit is set by software to clear the PLLSAI2RDYF flag.  
 0: No effect  
 1: Clear PLLSAI2RDYF flag
- 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 **LSIRDYC**: LSI ready interrupt clear  
 This bit is set by software to clear the LSIRDYF flag.  
 0: No effect  
 1: LSIRDYF cleared

### 6.4.10 AHB1 peripheral reset register (RCC\_AHB1RSTR)

Address offset: 0x28

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.	GFXMMURST	DMA2DRST	TSCRST
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	FLASHRST	Res.	Res.	Res.	Res.	Res.	DMAMUX1RST	DMA2RST	DMA1RST
			rw				rw						rw	rw	rw

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

- Bit 18 **GFXMMURST**: GFXMMU reset  
 Set and cleared by software  
 0: No effect  
 1: Reset GFXMMU

- Bit 17 **DMA2DRST**: DMA2D reset  
 Set and cleared by software  
 0: No effect  
 1: Reset DMA2D

- Bit 16 **TSCRST**: Touch Sensing Controller reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset TSC

Bits 15:13 Reserved, must be kept at reset value.

- Bit 12 **CRCRST**: CRC reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset CRC

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

Bit 8 **FLASHRST**: Flash memory interface reset

Set and cleared by software. This bit can be activated only when the Flash memory is in power down mode.

- 0: No effect
- 1: Reset Flash memory interface

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

Bit 2 **DMAMUX1RST**

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

### 6.4.11 AHB2 peripheral reset register (RCC\_AHB2RSTR)

Address offset: 0x2C

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.	SDMMC1RST	Res.	OSPIMRST	Res.	RNGRST	HASHRST	AESRST
													rW	rW	rW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DCMIRST	ADCRST	OTGFSRST	Res.	Res.	Res.	GPIOIRST	GPIOHRST	GPIOGRST	GPIOFIRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
	rW	rW	rW				rW	rW	rW	rW	rW	rW	rW	rW	rW

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

Bit 22 **SDMMC1RST**: SDMMC1 reset

Set and cleared by software.

- 0: No effect
- 1: Reset SDMMC1

Bit 21 Reserved, must be kept at reset value.

Bit 20 **OSPIMRST**: OctoSPI IO manager reset

Set and cleared by software.

- 0: No effect
- 1: Reset OctoSPI IO manager

- Bit 18 **RNGRST**: Random number generator reset  
Set and cleared by software.  
0: No effect  
1: Reset RNG
- Bit 17 **HASHRST**: Hash reset  
Set and cleared by software.  
0: No effect  
1: Reset HASH
- Bit 16 **AESRST**: AES hardware accelerator reset  
Set and cleared by software.  
0: No effect  
1: Reset AES
- Bit 15 Reserved, must be kept at reset value.
- Bit 14 **DCMIRST**: Digital Camera Interface reset  
Set and cleared by software  
0: No effect  
1: Reset DCMI interface
- Bit 13 **ADCRST**: ADC reset  
Set and cleared by software.  
0: No effect  
1: Reset ADC interface
- Bit 12 **OTGFSRST**: USB OTG FS reset  
Set and cleared by software.  
0: No effect  
1: Reset USB OTG FS
- Bits 11:9 Reserved, must be kept at reset value.
- Bit 8 **GPIIRST**: IO port I reset  
Set and cleared by software  
0: No effect  
1: Reset IO port I
- Bit 7 **GPIOHRST**: IO port H reset  
Set and cleared by software.  
0: No effect  
1: Reset IO port H
- Bit 6 **GPIOGRST**: IO port G reset  
Set and cleared by software.  
0: No effect  
1: Reset IO port G
- Bit 5 **GPIOFRST**: IO port F reset  
Set and cleared by software.  
0: No effect  
1: Reset IO port F

- 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  
Set and cleared by software.  
0: No effect  
1: Reset IO port D
- 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

### 6.4.12 AHB3 peripheral reset register (RCC\_AHB3RSTR)

Address offset: 0x30

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.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OSPI2 RST	OSPI1R ST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCR ST
						rw	rw								rw

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

- Bit 9 **OSPI2RST**: OctoSPI2 memory interface reset  
Set and cleared by software.  
0: No effect  
1: Reset OctoSPI2

Bit 8 **OSPI1RST**: OctoSPI1 memory interface reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset OctoSPI1

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **FMCRST**: Flexible memory controller reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset FMC

### 6.4.13 APB1 peripheral reset register 1 (RCC\_APB1RSTR1)

Address offset: 0x38

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1RST	OPAMP RST	DAC1 RST	PWRRST	Res.	Res.	CAN1RST	CRSRSRST	I2C3RST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Res.
rw	rw	rw	rw			rw		rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3RST	SPI2RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
rw	rw									rw	rw	rw	rw	rw	rw

Bit 31 **LPTIM1RST**: Low Power Timer 1 reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset LPTIM1

Bit 30 **OPAMP RST**: OPAMP interface reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset OPAMP interface

Bit 29 **DAC1RST**: DAC1 interface reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset DAC1 interface

Bit 28 **PWRRST**: Power interface reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset PWR

Bit 27:26 Reserved, must be kept at reset value.

Bit 25 **CAN1RST**: CAN1 reset  
 Set and reset by software.  
 0: No effect  
 1: Reset the CAN1



- 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: Reset I2C3
- Bit 22 **I2C2RST**: I2C2 reset  
Set and cleared by software.  
0: No effect  
1: Reset I2C2
- Bit 21 **I2C1RST**: I2C1 reset  
Set and cleared by software.  
0: No effect  
1: Reset I2C1
- Bit 20 **UART5RST**: UART5 reset  
Set and cleared by software.  
0: No effect  
1: Reset UART5
- Bit 19 **UART4RST**: UART4 reset  
Set and cleared by software.  
0: No effect  
1: Reset UART4
- Bit 18 **USART3RST**: USART3 reset  
Set and cleared by software.  
0: No effect  
1: Reset USART3
- Bit 17 **USART2RST**: USART2 reset  
Set and cleared by software.  
0: No effect  
1: Reset USART2
- Bit 16 Reserved, must be kept at reset value.
- Bit 15 **SPI3RST**: SPI3 reset  
Set and cleared by software.  
0: No effect  
1: Reset SPI3
- Bit 14 **SPI2RST**: SPI2 reset  
Set and cleared by software.  
0: No effect  
1: Reset SPI2
- Bits 13:6 Reserved, must be kept at reset value.
- Bit 5 **TIM7RST**: TIM7 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM7

- Bit 4 **TIM6RST**: TIM6 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM6
- Bit 3 **TIM5RST**: TIM5 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM5
- Bit 2 **TIM4RST**: TIM3 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM3
- Bit 1 **TIM3RST**: TIM3 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM3
- Bit 0 **TIM2RST**: TIM2 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM2

#### 6.4.14 APB1 peripheral reset register 2 (RCC\_APB1RSTR2)

Address offset: 0x3C

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.	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.	LPTIM2 RST	Res.	Res.	Res.	I2C4 RST	LPUART 1RST
										rw				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 4:2 Reserved, must be kept at reset value.

- Bit 1 **I2C4RST**: I2C4 reset  
Set and cleared by software  
0: No effect  
1: Reset I2C4

- Bit 0 **LPUART1RST**: Low-power UART 1 reset  
Set and cleared by software.  
0: No effect  
1: Reset LPUART1

### 6.4.15 APB2 peripheral reset register (RCC\_APB2RSTR)

Address offset: 0x40

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.	DSIRST	LTDCRST	Res.	DFSDM1RST	Res.	SAI2RST	SAI1RST	Res.	Res.	TIM17RST	TIM16RST	TIM15RST
				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.	USART1RST	TIM8RST	SPI1RST	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGRST
	rw	rw	rw	rw											rw

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

- Bit 27 **DSIRST**: DSI reset  
Set and cleared by software.  
0: No effect  
1: Reset DSI
- Bit 26 **LTDCRST**: LCD-TFT reset  
Set and cleared by software.  
0: No effect  
1: Reset LCD-TFT
- Bit 25 Reserved, must be kept at reset value.
- Bit 24 **DFSDM1RST**: Digital filters for sigma-delta modulators (DFSDM1) reset  
Set and cleared by software.  
0: No effect  
1: Reset DFSDM1
- Bit 23 Reserved, must be kept at reset value.
- Bit 22 **SAI2RST**: Serial audio interface 2 (SAI2) reset  
Set and cleared by software.  
0: No effect  
1: Reset SAI2

- 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
- Bit 16 **TIM15RST**: TIM15 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM15 timer
- Bit 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 **TIM8RST**: TIM8 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM8 timer
- Bit 12 **SPI1RST**: SPI1 reset  
Set and cleared by software.  
0: No effect  
1: Reset SPI1
- Bit 11 **TIM1RST**: TIM1 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM1 timer
- Bits 10:1 Reserved, must be kept at reset value.
- Bit 0 **SYSCFGRST**: SYSCFG + COMP + VREFBUF reset  
0: No effect  
1: Reset SYSCFG + COMP + VREFBUF

### 6.4.16 AHB1 peripheral clock enable register (RCC\_AHB1ENR)

Address offset: 0x48

Reset value: 0x0000 0100

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 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.	GFXM MUEN	DMA2D EN	TSCE N
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCEN	Res.	Res.	Res.	FLASH EN	Res.	Res.	Res.	Res.	Res.	DMAM UX1EN	DMA2E N	DMA1 EN
			rw				rw						rw	rw	rw

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

Bit 18 **GFXMMUEN**: Graphic MMU clock enable

- Set and reset by software
- 0: GFXMMU clock disabled
- 1: GFXMMU clock enabled

Bit 17 **DMA2DEN**: DMA2D clock enable

- Set and cleared by software
- 0: DMA2D clock disabled
- 1: DMA2D clock enabled

Bit 16 **TSCEEN**: Touch Sensing Controller clock enable

- Set and cleared by software.
- 0: TSC clock disable
- 1: TSC clock enable

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCEN**: CRC clock enable

- Set and cleared by software.
- 0: CRC clock disable
- 1: CRC clock enable

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

Bit 8 **FLASHEN**: Flash memory interface clock enable

- Set and cleared by software. This bit can be disabled only when the Flash is in power down mode.
- 0: Flash memory interface clock disable
- 1: Flash memory interface clock enable

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

Bit 2 **DMAMUX1EN**: DMAMUX1 clock enable

Set and reset by software.  
 0: DMAMUX1 clock disabled  
 1: DMAMUX1 clock enabled

Bit 1 **DMA2EN**: DMA2 clock enable

Set and cleared by software.  
 0: DMA2 clock disable  
 1: DMA2 clock enable

Bit 0 **DMA1EN**: DMA1 clock enable

Set and cleared by software.  
 0: DMA1 clock disable  
 1: DMA1 clock enable

### 6.4.17 AHB2 peripheral clock enable register (RCC\_AHB2ENR)

Address offset: 0x4C

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 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.	SDMMC1EN	Res.	OSPIMEN	Res.	RNGEN	HASHEN	AESEN
									rw		rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DCMIEN	ADCEN	OTGFSEN	Res.	Res.	Res.	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 22 **SDMMC1EN**: SDMMC1 clock enable

Set and cleared by software.  
 0: SDMMC1 clock disabled  
 1: SDMMC1 clock enabled

Bit 21 Reserved, must be kept at reset value.

Bit 20 **OSPIMEN**: OctoSPI IO manager clock enable

Set and cleared by software.  
 0: OctoSPI IO manager clock disabled  
 1: OctoSPI IO manager clock enabled

Bit 19 Reserved, must be kept at reset value.

Bit 18 **RNGEN**: Random Number Generator clock enable

Set and cleared by software.  
 0: Random Number Generator clock disabled  
 1: Random Number Generator clock enabled

- Bit 17 **HASHEN**: HASH clock enable  
Set and cleared by software  
0: HASH clock disabled  
1: HASH clock enabled
- Bit 16 **AESEN**: AES accelerator clock enable  
Set and cleared by software.  
0: AES clock disabled  
1: AES clock enabled
- Bit 15 Reserved, must be kept at reset value.
- Bit 14 **DCMIEN**: DCMI clock enable  
Set and cleared by software  
0: DCMI clock disabled  
1: DCMI clock enabled
- Bit 13 **ADCEN**: ADC clock enable  
Set and cleared by software.  
0: ADC clock disabled  
1: ADC clock enabled
- Bit 12 **OTGFSEN**: OTG full speed clock enable  
Set and cleared by software.  
0: USB OTG full speed clock disabled  
1: USB OTG full speed clock enabled
- Bits 11:9 Reserved, must be kept at reset value.
- Bit 8 **GPIOIEN**: IO port I clock enable  
Set and cleared by software  
0: IO port I clock disabled  
1: IO port I clock enabled
- Bit 7 **GPIOHEN**: IO port H clock enable  
Set and cleared by software.  
0: IO port H clock disabled  
1: IO port H clock enabled
- Bit 6 **GPIOGEN**: IO port G clock enable  
Set and cleared by software.  
0: IO port G clock disabled  
1: IO port G clock enabled
- Bit 5 **GPIOFEN**: IO port F clock enable  
Set and cleared by software.  
0: IO port F clock disabled  
1: IO port F clock enabled
- Bit 4 **GPIOEEN**: IO port E clock enable  
Set and cleared by software.  
0: IO port E clock disabled  
1: IO port E clock enabled

- Bit 3 **GPIODEN**: IO port D clock enable  
Set and cleared by software.  
0: IO port D clock disabled  
1: IO port D clock enabled
- Bit 2 **GPIOCEN**: IO port C clock enable  
Set and cleared by software.  
0: IO port C clock disabled  
1: IO port C clock enabled
- Bit 1 **GPIOBEN**: IO port B clock enable  
Set and cleared by software.  
0: IO port B clock disabled  
1: IO port B clock enabled
- Bit 0 **GPIOAEN**: IO port A clock enable  
Set and cleared by software.  
0: IO port A clock disabled  
1: IO port A clock enabled

### 6.4.18 AHB3 peripheral clock enable register(RCC\_AHB3ENR)

Address offset: 0x50

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 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.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OSPI2 EN	OSPI1E N	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCE N
						rw	rw								rw

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

- Bit 9 **OSPI2EN**: OctoSPI2 memory interface clock enable  
Set and cleared by software.  
0: OctoSPI2 clock disable  
1: OctoSPI2 clock enable



Bit 8 **OSPI1EN**: OctoSPI1 memory interface clock enable  
 Set and cleared by software.  
 0: OctoSPI1 clock disable  
 1: OctoSPI1 clock enable

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **FMCEN**: Flexible memory controller clock enable  
 Set and cleared by software.  
 0: FMC clock disable  
 1: FMC clock enable

### 6.4.19 APB1 peripheral clock enable register 1 (RCC\_APB1ENR1)

Address: 0x58

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 is not supported.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1 EN	OPAMP EN	DAC1 EN	PWREN	Res.	Res.	CAN1EN	CRSEN	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.
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
SPI3EN	SPI2EN	Res.	Res.	WWDGEN	RTCPBEN	Res.	Res.	Res.	Res.	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
rW	rW			rs	rW					rW	rW	rW	rW	rW	rW

Bit 31 **LPTIM1EN**: Low power timer 1 clock enable  
 Set and cleared by software.  
 0: LPTIM1 clock disabled  
 1: LPTIM1 clock enabled

Bit 30 **OPAMPEN**: OPAMP interface clock enable  
 Set and cleared by software.  
 0: OPAMP interface clock disabled  
 1: OPAMP interface clock enabled

Bit 29 **DAC1EN**: DAC1 interface clock enable  
 Set and cleared by software.  
 0: DAC1 interface clock disabled  
 1: DAC1 interface clock enabled

Bit 28 **PWREN**: Power interface clock enable  
 Set and cleared by software.  
 0: Power interface clock disabled  
 1: Power interface clock enabled

Bit 27:26 Reserved, must be kept at reset value.

- Bit 25 **CAN1EN**: CAN1 clock enable  
Set and cleared by software.  
0: CAN1 clock disabled  
1: CAN1 clock enabled
- Bit 24 **CRSEN**: Clock Recovery System clock enable  
Set and cleared by software  
0: CRS clock disabled  
1: CRS clock enabled
- Bit 23 **I2C3EN**: I2C3 clock enable  
Set and cleared by software.  
0: I2C3 clock disabled  
1: I2C3 clock enabled
- Bit 22 **I2C2EN**: I2C2 clock enable  
Set and cleared by software.  
0: I2C2 clock disabled  
1: I2C2 clock enabled
- Bit 21 **I2C1EN**: I2C1 clock enable  
Set and cleared by software.  
0: I2C1 clock disabled  
1: I2C1 clock enabled
- Bit 20 **UART5EN**: UART5 clock enable  
Set and cleared by software.  
0: UART5 clock disabled  
1: UART5 clock enabled
- Bit 19 **UART4EN**: UART4 clock enable  
Set and cleared by software.  
0: UART4 clock disabled  
1: UART4 clock enabled
- Bit 18 **USART3EN**: USART3 clock enable  
Set and cleared by software.  
0: USART3 clock disabled  
1: USART3 clock enabled
- Bit 17 **USART2EN**: USART2 clock enable  
Set and cleared by software.  
0: USART2 clock disabled  
1: USART2 clock enabled
- Bit 16 Reserved, must be kept at reset value.
- Bit 15 **SPI3EN**: SPI3 clock enable  
Set and cleared by software.  
0: SPI3 clock disabled  
1: SPI3 clock enabled
- Bit 14 **SPI2EN**: SPI2 clock enable  
Set and cleared by software.  
0: SPI2 clock disabled  
1: SPI2 clock enabled
- Bits 13:12 Reserved, must be kept at reset value.

- Bit 11 **WWDGEN**: Window watchdog clock enable  
Set 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  
1: Window watchdog clock enabled
- Bit 10 **RTCAPBEN**: RTC APB clock enable  
Set and cleared by software  
0: RTC APB clock disabled  
1: RTC APB clock enabled
- Bits 9:6 Reserved, must be kept at reset value.
- Bit 5 **TIM7EN**: TIM7 timer clock enable  
Set and cleared by software.  
0: TIM7 clock disabled  
1: TIM7 clock enabled
- Bit 4 **TIM6EN**: TIM6 timer clock enable  
Set and cleared by software.  
0: TIM6 clock disabled  
1: TIM6 clock enabled
- Bit 3 **TIM5EN**: TIM5 timer clock enable  
Set and cleared by software.  
0: TIM5 clock disabled  
1: TIM5 clock enabled
- Bit 2 **TIM4EN**: TIM4 timer clock enable  
Set and cleared by software.  
0: TIM4 clock disabled  
1: TIM4 clock enabled
- Bit 1 **TIM3EN**: TIM3 timer clock enable  
Set and cleared by software.  
0: TIM3 clock disabled  
1: TIM3 clock enabled
- Bit 0 **TIM2EN**: TIM2 timer clock enable  
Set and cleared by software.  
0: TIM2 clock disabled  
1: TIM2 clock enabled

#### 6.4.20 APB1 peripheral clock enable register 2 (RCC\_APB1ENR2)

Address offset: 0x5C

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 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.	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.	LPTIM2 EN	Res.	Res.	Res.	I2C4EN	LPUAR T1EN
														rw	rw

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

Bit 5 **LPTIM2EN** Low power timer 2 clock enable

Set and cleared by software.

0: LPTIM2 clock disable

1: LPTIM2 clock enable

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **I2C4EN**: I2C4 clock enable

Set and cleared by software

0: I2C4 clock disabled

1: I2C4 clock enabled

Bit 0 **LPUART1EN**: Low power UART 1 clock enable

Set and cleared by software.

0: LPUART1 clock disable

1: LPUART1 clock enable

### 6.4.21 APB2 peripheral clock enable register (RCC\_APB2ENR)

Address: 0x60

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 is not supported.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DSIEN	LTDCE N	Res.	DFSD M1EN	Res.	SAI2E N	SAI1E N	Res.	Res.	TIM17E N	TIM16E N	TIM15E N
				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.	USART 1EN	TIM8E N	SP11E N	TIM1E N	Res.	Res.	Res.	FWEN	Res.	Res.	Res.	Res.	Res.	Res.	SYSCF GEN
	rw	rw	rw	rw				rs							rw

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

Bit 27 **DSIEN**: DSI clock enable

Set and cleared by software.

0: DSI clock disabled

1: DSI clock enable

Bit 26 **LTDCEN**: LCD-TFT clock enable

Set and cleared by software.

0: LTDC clock disabled

1: LTDC clock enable

Bit 25 Reserved, must be kept at reset value.

Bit 24 **DFSDM1EN**: DFSDM1 timer clock enable

Set and cleared by software.

0: DFSDM1 clock disabled

1: DFSDM1 clock enabled

Bit 23 Reserved, must be kept at reset value.

Bit 22 **SAI2EN**: SAI2 clock enable

Set and cleared by software.

0: SAI2 clock disabled

1: SAI2 clock enabled

Bit 21 **SAI1EN**: SAI1 clock enable

Set and cleared by software.

0: SAI1 clock disabled

1: SAI1 clock enabled

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TIM17EN**: TIM17 timer clock enable

Set and cleared by software.

0: TIM17 timer clock disabled

1: TIM17 timer clock enabled

- Bit 17 **TIM16EN**: TIM16 timer clock enable  
Set and cleared by software.  
0: TIM16 timer clock disabled  
1: TIM16 timer clock enabled
- Bit 16 **TIM15EN**: TIM15 timer clock enable  
Set and cleared by software.  
0: TIM15 timer clock disabled  
1: TIM15 timer clock enabled
- Bit 15 Reserved, must be kept at reset value.
- Bit 14 **USART1EN**: USART1 clock enable  
Set and cleared by software.  
0: USART1 clock disabled  
1: USART1 clock enabled
- Bit 13 **TIM8EN**: TIM8 timer clock enable  
Set and cleared by software.  
0: TIM8 timer clock disabled  
1: TIM8 timer clock enabled
- Bit 12 **SPI1EN**: SPI1 clock enable  
Set and cleared by software.  
0: SPI1 clock disabled  
1: SPI1 clock enabled
- Bit 11 **TIM1EN**: TIM1 timer clock enable  
Set and cleared by software.  
0: TIM1 timer clock disabled  
1: TIM1P timer clock enabled
- Bits 10:8 Reserved, must be kept at reset value.
- Bit 7 **FWEN**: Firewall clock enable  
Set by software, reset by hardware. Software can only write 1. A write at 0 has no effect.  
0: Firewall clock disabled  
1: Firewall clock enabled
- Bits 6:1 Reserved, must be kept at reset value.
- Bit 0 **SYSCFGEN**: SYSCFG + COMP + VREFBUF clock enable  
Set and cleared by software.  
0: SYSCFG + COMP + VREFBUF clock disabled  
1: SYSCFG + COMP + VREFBUF clock enabled

#### 6.4.22 AHB1 peripheral clocks enable in Sleep and Stop modes register (RCC\_AHB1SMENR)

Address offset: 0x68

Reset value: 0x0007 1307

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.	GFXM MUSM EN	DMA2D SMEN	TSCS MEN
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCSMEN	Res.	Res.	SRAM1 SMEN	FLASH SMEN	Res.	Res.	Res.	Res.	Res.	DMAM UX1S MEN	DMA2S MEN	DMA1 SMEN
			rw			rw	rw							rw	rw

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

Bit 18 **GFXMMUSMEN**: GFXMMU clock enable during Sleep and Stop modes.  
 Set and cleared by software  
 0: GFXMMU clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: GFXMMU clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 17 **DMA2DSMEN**: DMA2D clock enable during Sleep and Stop modes  
 Set and cleared by software  
 0: DMA2D clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: DMA2D clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 16 **TSCSMEN**: Touch Sensing Controller clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: TSC clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: TSC clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCSMEN**: CRC clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: CRC clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: CRC clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **SRAM1SMEN**: SRAM1 interface clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: SRAM1 interface clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: SRAM1 interface clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 8 **FLASHSMEN**: Flash memory interface clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: Flash memory interface clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: Flash memory interface clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

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

Bit 2 **DMAMUX1SMEN**: DMAMUX1 clock enable during Sleep and Stop modes.  
 Set and cleared by software.  
 0: DMAMUX1 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: DMAMUX1 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 1 **DMA2SMEN**: DMA2 clocks enable during Sleep and Stop modes  
 Set and cleared by software during Sleep mode.  
 0: DMA2 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: DMA2 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 0 **DMA1SMEN**: DMA1 clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: DMA1 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: DMA1 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1. This register only configures the clock gating, not the clock source itself. Most of the peripherals are clocked by a single clock (AHB or APB clock), which is always disabled in Stop mode. In this case setting the bit has no effect in Stop mode.

### 6.4.23 AHB2 peripheral clocks enable in Sleep and Stop modes register (RCC\_AHB2SMENR)

Address offset: 0x6C

Reset value: 0x0057 77FF

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.	SDMMC1SMEN	Res.	OSPIMSMEN	Res.	RNGSMEN	HASHSMEN	AESSMEN
									rw				rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DCMISMEN	ADCSMEN	OTGFSSMEN	Res.	SRAM3SMEN	SRAM2SMEN	GPIOISMEN	GPIOHSMEN	GPIOGSMEN	GPIOFSMEN	GPIOESMEN	GPIODSMEN	GPIOCSMEN	GPIOBSMEN	GPIOASMEN
	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 **SDMMC1SMEN**: SDMMC1 clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: SDMMC1 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: SDMMC1 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 21 Reserved, must be kept at reset value.

Bit 20 **OSPIMSMEN**: OctoSPI IO manager clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: OCTOSPIM clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: OCTOSPIM clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 19 Reserved, must be kept at reset value.



- Bit 18 **RNGSMEN**: Random Number Generator clocks enable during Sleep and Stop modes  
Set and cleared by software.  
0: Random Number Generator clocks disabled by the clock gating during Sleep and Stop modes  
1: Random Number Generator clocks enabled by the clock gating during Sleep and Stop modes
- Bit 17 **HASHSMEN**: HASH clock enable during Sleep and Stop modes  
Set and cleared by software  
0: HASH clocks disabled by the clock gating<sup>(2)</sup> during Sleep and Stop modes  
1: HASH clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 16 **AESSMEN**: AES accelerator clocks enable during Sleep and Stop modes  
Set and cleared by software.  
0: AES clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: AES clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bits 15 Reserved, must be kept at reset value.
- Bit 14 **DCMISMEN**: DCMI clock enable during Sleep and Stop modes  
Set and cleared by software  
0: DCMI clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: DCMI clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 13 **ADCSMEN**: ADC clocks enable during Sleep and Stop modes  
Set and cleared by software.  
0: ADC clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: ADC clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 12 **OTGFSSMEN**: OTG full speed clocks enable during Sleep and Stop modes  
Set and cleared by software.  
0: USB OTG full speed clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: USB OTG full speed clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bits 11 Reserved, must be kept at reset value.
- Bit 10 **SRAM3SMEN**: SRAM2 interface clocks enable during Sleep and Stop modes  
Set and cleared by software.  
0: SRAM3 interface clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: SRAM32 interface clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 9 **SRAM2SMEN**: SRAM2 interface clocks enable during Sleep and Stop modes  
Set and cleared by software.  
0: SRAM2 interface clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: SRAM2 interface clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 8 **GPIOISMEN**: IO port I clocks enable during Sleep and Stop modes  
Set and cleared by software  
0: IO port I clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: IO port I clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 7 **GPIOHSMEN**: IO port H clocks enable during Sleep and Stop modes  
Set and cleared by software.  
0: IO port H clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: IO port H clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

- Bit 6 **GPIOGSMEN**: IO port G clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: IO port G clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: IO port G clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 5 **GPIOFSMEN**: IO port F clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: IO port F clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: IO port F clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 4 **GPIOESMEN**: IO port E clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: IO port E clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: IO port E clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 3 **GPIODSMEN**: IO port D clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: IO port D clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: IO port D clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 2 **GPIOCSMEN**: IO port C clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: IO port C clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: IO port C clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 1 **GPIOBSMEN**: IO port B clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: IO port B clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: IO port B clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 0 **GPIOASMEN**: IO port A clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: IO port A clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: IO port A clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1. This register only configures the clock gating, not the clock source itself. Most of the peripherals are clocked by a single clock (AHB or APB clock), which is always disabled in Stop mode. In this case setting the bit has no effect in Stop mode.
2. This register only configures the clock gating, not the clock source itself. Most of the peripherals are clocked by a single clock (AHB or APB clock), which is always disabled in Stop mode. In this case setting the bit has no effect in Stop mode.

### 6.4.24 AHB3 peripheral clocks enable in Sleep and Stop modes register (RCC\_AHB3SMENR)

Address offset: 0x70

Reset value: 0x00000 0301

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.	Res.	Res.	OCTOS PI2	OSPI1S MEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCS MEN
						rw	rw								rw



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

Bit 9 **OCTOSPI2**: OctoSPI2 memory interface clocks enable during Sleep and Stop modes  
Set and cleared by software.

0: OctoSPI2 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: OctoSPI2 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 8 **OSPI1SMEN**: OctoSPI1 memory interface clocks enable during Sleep and Stop modes  
Set and cleared by software.

0: OctoSPI1 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: OctoSPI1 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **FMCSMEN**: Flexible memory controller clocks enable during Sleep and Stop modes  
Set and cleared by software.

0: FMC clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: FMC clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1. This register only configures the clock gating, not the clock source itself. Most of the peripherals are clocked by a single clock (AHB or APB clock), which is always disabled in Stop mode. In this case setting the bit has no effect in Stop mode.

### 6.4.25 APB1 peripheral clocks enable in Sleep and Stop modes register 1 (RCC\_APB1SMENR1)

Address: 0x78

Reset value: 0xF3FECC3F

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	OPAMP SMEN	DAC1 SMEN	PWRS MEN	Res.	Res.	CAN1S MEN	CRSS MEN	I2C3S MEN	I2C2S MEN	I2C1S MEN	UART5S MEN	UART4S MEN	USART3 SMEN	USART2 SMEN	Res.
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
SPI3S MEN	SPI2S MEN	Res.	Res.	WWDG SMEN	RTCA PBSM EN	Res.	Res.	Res.	Res.	TIM7S MEN	TIM6SM EN	TIM5SM EN	TIM4SM EN	TIM3SM EN	TIM2S MEN
rw	rw			rw	rw					rw	rw	rw	rw	rw	rw

Bit 31 **LPTIM1SMEN**: Low power timer 1 clocks enable during Sleep and Stop modes  
Set and cleared by software.

0: LPTIM1 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: LPTIM1 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 30 **OPAMP SMEN**: OPAMP interface clocks enable during Sleep and Stop modes  
Set and cleared by software.

0: OPAMP interface clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: OPAMP interface clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 29 **DAC1SMEN**: DAC1 interface clocks enable during Sleep and Stop modes  
Set and cleared by software.

0: DAC1 interface clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: DAC1 interface clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

- Bit 28 **PWRSMEN**: Power interface clocks enable during Sleep and Stop modes  
Set and cleared by software.  
0: Power interface clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: Power interface clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 27 Reserved, must be kept at reset value.
- Bit 25 **CAN1SMEN**: CAN1 clocks enable during Sleep and Stop modes  
Set and cleared by software.  
0: CAN1 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: CAN1 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 24 **CRSSMEN**: CRS clock enable during Sleep and Stop modes  
Set and cleared by software.  
0: CRS clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: CRS clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 23 **I2C3SMEN**: I2C3 clocks enable during Sleep and Stop modes  
Set and cleared by software.  
0: I2C3 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: I2C3 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 22 **I2C2SMEN**: I2C2 clocks enable during Sleep and Stop modes  
Set and cleared by software.  
0: I2C2 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: I2C2 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 21 **I2C1SMEN**: I2C1 clocks enable during Sleep and Stop modes  
Set and cleared by software.  
0: I2C1 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: I2C1 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 20 **UART5SMEN**: UART5 clocks enable during Sleep and Stop modes  
Set and cleared by software.  
0: UART5 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: UART5 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 19 **UART4SMEN**: UART4 clocks enable during Sleep and Stop modes  
Set and cleared by software.  
0: UART4 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: UART4 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 18 **USART3SMEN**: USART3 clocks enable during Sleep and Stop modes  
Set and cleared by software.  
0: USART3 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: USART3 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 17 **USART2SMEN**: USART2 clocks enable during Sleep and Stop modes  
Set and cleared by software.  
0: USART2 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: USART2 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 16 Reserved, must be kept at reset value.
- Bit 15 **SPI3SMEN**: SPI3 clocks enable during Sleep and Stop modes  
Set and cleared by software.  
0: SPI3 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
1: SPI3 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 14 **SPI2SMEN**: SPI2 clocks enable during Sleep and Stop modes

Set and cleared by software.

0: SPI2 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: SPI2 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

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

Bit 11 **WWDGSMEN**: Window watchdog clocks enable during Sleep and Stop modes

Set and cleared by software. This bit is forced to '1' by hardware when the hardware WWDG option is activated.

0: Window watchdog clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: Window watchdog clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 10 **RTCAPBSMEN**: RTC APB clock enable during Sleep and Stop modes

Set and cleared by software

0: RTC APB clock disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: RTC APB clock enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

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

Bit 5 **TIM7SMEN**: TIM7 timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: TIM7 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: TIM7 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 4 **TIM6SMEN**: TIM6 timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: TIM6 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: TIM6 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 3 **TIM5SMEN**: TIM5 timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: TIM5 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: TIM5 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 2 **TIM4SMEN**: TIM4 timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: TIM4 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: TIM4 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 1 **TIM3SMEN**: TIM3 timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: TIM3 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: TIM3 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 0 **TIM2SMEN**: TIM2 timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: TIM2 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: TIM2 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1. This register only configures the clock gating, not the clock source itself. Most of the peripherals are clocked by a single clock (AHB or APB clock), which is always disabled in Stop mode. In this case setting the bit has no effect in Stop mode.

#### 6.4.26 APB1 peripheral clocks enable in Sleep and Stop modes register 2 (RCC\_APB1SMENR2)

Address offset: 0x7C

Reset value: 0x0000 0023

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.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2SMEN	Res.	Res.	Res.	I2C4SMEN	LPUART1SMEN
										rw				rw	rw

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

Bit 5 **LPTIM2SMEN** Low power timer 2 clocks enable during Sleep and Stop modes

Set and cleared by software.

0: LPTIM2 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: LPTIM2 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

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

Bit 1 **I2C4SMEN**: I2C4 clocks enable during Sleep and Stop modes

Set and cleared by software

0: I2C4 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: I2C4 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 0 **LPUART1SMEN**: Low power UART 1 clocks enable during Sleep and Stop modes

Set and cleared by software.

0: LPUART1 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: LPUART1 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1. This register only configures the clock gating, not the clock source itself. Most of the peripherals are clocked by a single clock (AHB or APB clock), which is always disabled in Stop mode. In this case setting the bit has no effect in Stop mode.

### 6.4.27 APB2 peripheral clocks enable in Sleep and Stop modes register (RCC\_APB2SMENR)

Address: 0x80

Reset value: 0x0D67 7801

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.	DSISMEN	LTDCSMEN	Res.	DFSDM1SMEN	Res.	SAI2SMEN	SAI1SMEN	Res.	Res.	TIM17SMEN	TIM16SMEN	TIM15SMEN
				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.	USART1SMEN	TIM8SMEN	SPI1SMEN	TIM1SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGSMEN
	rw	rw	rw	rw											rw

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

Bit 27 **DSISMEN**: DSI clocks enable during Sleep and Stop modes

Set and cleared by software.

0: DSI clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: DSI clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 26 **LTDCSMEN**: LCD-TFT timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: LCD-TFT clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: LCD-TFT clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 25 Reserved, must be kept at reset value.

Bit 24 **DFSDM1SMEN**: DFSDM1 timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: DFSDM1 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: DFSDM1 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 23 Reserved, must be kept at reset value.

Bit 22 **SAI2SMEN**: SAI2 clocks enable during Sleep and Stop modes

Set and cleared by software.

0: SAI2 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: SAI2 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bit 21 **SAI1SMEN**: SAI1 clocks enable during Sleep and Stop modes

Set and cleared by software.

0: SAI1 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: SAI1 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TIM17SMEN**: TIM17 timer clocks enable during Sleep and Stop modes

Set and cleared by software.

0: TIM17 timer clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1: TIM17 timer clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

- Bit 17 **TIM16SMEN**: TIM16 timer clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: TIM16 timer clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: TIM16 timer clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 16 **TIM15SMEN**: TIM15 timer clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: TIM15 timer clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: TIM15 timer clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 15 Reserved, must be kept at reset value.
- Bit 14 **USART1SMEN**: USART1 clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: USART1 clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: USART1 clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 13 **TIM8SMEN**: TIM8 timer clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: TIM8 timer clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: TIM8 timer clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bit 12 **SPI1SMEN**: SPI1 clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: SPI1 clocks disabled by the clock gating during<sup>(1)</sup> Sleep and Stop modes  
 1: SPI1 clocks enabled by the clock gating during<sup>(1)</sup> Sleep and Stop modes
- Bit 11 **TIM1SMEN**: TIM1 timer clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: TIM1 timer clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: TIM1P timer clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes
- Bits 10:1 Reserved, must be kept at reset value.
- Bit 0 **SYSCFGSMEN**: SYSCFG + COMP + VREFBUF clocks enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: SYSCFG + COMP + VREFBUF clocks disabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes  
 1: SYSCFG + COMP + VREFBUF clocks enabled by the clock gating<sup>(1)</sup> during Sleep and Stop modes

1. This register only configures the clock gating, not the clock source itself. Most of the peripherals are clocked by a single clock (AHB or APB clock), which is always disabled in Stop mode. In this case setting the bit has no effect in Stop mode.

#### 6.4.28 Peripherals independent clock configuration register (RCC\_CCIPR)

Address: 0x88

Reset value: 0x0000 0000

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



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ADCSEL[1:0]		CLK48SEL[1:0]		Res.				LPTIM2SEL[1:0]		LPTIM1SEL[1:0]		I2C3SEL[1:0]	
		rw	rw	rw	rw					rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2C2SEL[1:0]		I2C1SEL[1:0]		LPUART1SEL[1:0]		UART5SEL[1:0]		UART4SEL[1:0]		USART3SEL[1:0]		USART2SEL[1:0]		USART1SEL[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **ADCSEL[1:0]**: ADCs 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 (PLLADC1CLK) selected as ADC clock
- 10: Reserved
- 11: System clock selected as ADCs 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 OTG FS, RNG and SDMMC.

- 00: HSI48 clock selected as 48 MHz clock
- 01: PLLSAI1 “Q” clock (PLL48M2CLK) selected as 48 MHz clock
- 10: PLL “Q” clock (PLL48M1CLK) selected as 48 MHz clock
- 11: MSI clock selected as 48 MHz clock

Bits 22:25 Reserved, must be kept at reset value.

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 **I2C2SEL[1:0]**: I2C2 clock source selection  
These bits are set and cleared by software to select the I2C2 clock source.  
00: PCLK selected as I2C2 clock  
01: System clock (SYSCLK) selected as I2C2 clock  
10: HSI16 clock selected as I2C2 clock  
11: Reserved
- 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 9:8 **UART5SEL[1:0]**: UART5 clock source selection  
These bits are set and cleared by software to select the UART5 clock source.  
00: PCLK selected as UART5 clock  
01: System clock (SYSCLK) selected as UART5 clock  
10: HSI16 clock selected as UART5 clock  
11: LSE clock selected as UART5 clock
- Bits 7:6 **UART4SEL[1:0]**: UART4 clock source selection  
This bit is set and cleared by software to select the UART4 clock source.  
00: PCLK selected as UART4 clock  
01: System clock (SYSCLK) selected as UART4 clock  
10: HSI16 clock selected as UART4 clock  
11: LSE clock selected as UART4 clock
- Bits 5:4 **USART3SEL[1:0]**: USART3 clock source selection  
This bit is set and cleared by software to select the USART3 clock source.  
00: PCLK selected as USART3 clock  
01: System clock (SYSCLK) selected as USART3 clock  
10: HSI16 clock selected as USART3 clock  
11: LSE clock selected as USART3 clock
- Bits 3:2 **USART2SEL[1:0]**: USART2 clock source selection  
This bit is set and cleared by software to select the USART2 clock source.  
00: PCLK selected as USART2 clock  
01: System clock (SYSCLK) selected as USART2 clock  
10: HSI16 clock selected as USART2 clock  
11: LSE clock selected as USART2 clock
- 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

### 6.4.29 Backup domain control register (RCC\_BDCR)

Address offset: 0x90

Reset value: 0x0000 0000, reset by Backup domain Reset, except LSCOSEL, LSCOEN and BDRST which are reset only by Backup domain power-on 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 Backup domain control register (RCC\_BDCR) are outside of the V<sub>CORE</sub> domain. As a result, after Reset, these bits are write-protected and the DBP bit in the Section 5.4.1: Power control register 1 (PWR\_CR1) has to be set before these can be modified. Refer to Section 5.1.4: Battery backup domain on page 127 for further information. These bits (except LSCOSEL, LSCOEN and BDRST) are only reset after a Backup domain Reset (see Section 6.1.3: Backup domain reset). Any internal or external Reset will not have any effect on these bits.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	LSCOSEL	LSCOEN	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.	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	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: Reset 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's 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 in debug mode. 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 6 external low-speed oscillator clock cycles.

0: LSE oscillator not ready

1: LSE oscillator ready

**Bit 0 LSEON:** LSE oscillator enable

Set and cleared by software.

0: LSE oscillator OFF

1: LSE oscillator ON

### 6.4.30 Control/status register (RCC\_CSR)

Address: 0x94

Reset value: 0x0C00 0600, reset by system Reset, except reset flags by power Reset only.

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWWG RSTF	SFTRS TF	BORR STF	PINRS TF	OBLRS TF	FWRST F	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MSISRANGE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	LSIRDY	LSION
				rw	rw	rw	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 **FWRSTF**: Firewall reset flag  
 Set by hardware when a reset from the firewall occurs.  
 Cleared by writing to the RMVF bit.  
 0: No reset from the firewall occurred  
 1: Reset from the firewall occurred

Bit 23 **RMVF**: Remove reset flag  
 Set by software to clear the reset flags.  
 0: No effect  
 1: Clear the reset flags

Bits 22:12 Reserved, must be kept at reset value.

Bits 11:8 **MSISRANGE[3:0]** MSI range after Standby mode  
 Set by software to chose the MSI frequency at startup. This range is used after exiting Standby mode until MSIRGSEL is set. After a pad or a power-on reset, the range is always 4 MHz. MSISRANGE can be written only when MSIRGSEL = '1'.  
 0100: Range 4 around 1 MHz  
 0101: Range 5 around 2 MHz  
 0101: Range 6 around 4 MHz (reset value)  
 0111: Range 7 around 8 MHz  
 others: Reserved

*Note: Changing the MSISRANGE does not change the current MSI frequency.*

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **LSIRDY**: LSI oscillator ready  
 Set and cleared by hardware to indicate when the LSI oscillator is stable. After the LSION bit is cleared, LSIRDY goes low after 3 LSI oscillator clock cycles. This bit can be set even if LSION = 0 if the LSI is requested by the Clock Security System on LSE, by the Independent Watchdog or by the RTC.  
 0: LSI oscillator not ready  
 1: LSI oscillator ready

Bit 0 **LSION**: LSI oscillator enable  
 Set and cleared by software.  
 0: LSI oscillator OFF  
 1: LSI oscillator ON

### 6.4.31 Clock recovery RC register (RCC\_CRRCR)

Address: 0x98

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.	HSI48RDY	HSI48ON
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

### 6.4.32 Peripherals independent clock configuration register (RCC\_CCIPR2)

Address: 0x9C

Reset value: 0x0000 0000

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPISEL[1:0]		Res.	Res.	PLLSAI2DIVR[1:0]	
										rw				rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SDMMC SEL	Res.	DSISEL	Res.	SAI2SEL[2:0]			SAI1SEL[2:0]			ADFSDMSEL[1:0]		DFSD MSEL	I2C4SEL[1:0]	
	rw		rw		rw			rw			rw		rw	rw	

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **OSPISEL**: Octospi clock source selection

Set and reset by software.

00: system clock selected as OctoSPI kernel clock

01: MSI clock selected as OctoSPI kernel clock

10: PLL48M1CLK clock selected as OctoSPI kernel clock

11: reserved

Bits 19:18 Reserved, must be kept at reset value.

- Bits 17:16 **PLLSAI2DIVR**: division factor for LTDC clock  
Set and reset by software to control the frequency of LTDC clock.  
These bits can only be written when PLLSAI2 is disabled.  
LTDC clock frequency =  $f(\text{PLLSAI2\_R}) / \text{PLLSAI2DIVR}$  with  $2 \leq \text{PLLSAI2DIVR} \leq 16$   
00: PLLSAI2DIVR = /2  
01: PLLSAI2DIVR = /4  
10: PLLSAI2DIVR = /8  
11: PLLSAI2DIVR = /16
- Bit 15 Reserved, must be kept at reset value.
- Bit 14 **SDMMCSEL**: SDMMC clock selection  
Set and reset by software.  
This bit allows to select the SDMMC kernel clock source between PLLP clock (PLLSAI3CLK) or clock from internal multiplexor.  
It is recommended to change this bit only after reset and before enabling the SDMMC module.  
0: 48 MHz clock is selected as SDMMC kernel clock  
1: PLLSAI3CLK is selected as SDMMC kernel clock, used in case higher frequency than 48MHz is needed (for SDR50 mode).
- Bit 13 Reserved, must be kept at reset value.
- Bit 12 **DSISEL** clock selection  
Set and reset by software.  
This bit allows to select the DSI byte lane clock source between PLLDSICLK clock or clock from DSI-PHY.  
It is recommended to change this bit only after reset and before to enable the DSI module.  
0: DSI-PHY is selected as DSI byte lane clock source (usual case)  
1: PLLDSICLK is selected as DSI byte lane clock source, used in case DSI PLL and DSI-PHY are off (low-power mode).
- Bit 11 Reserved, must be kept at reset value.
- Bits 10:8 **SAI2SEL**: SAI2 clock source selection  
Set and reset by software.  
If the selected clock is the external clock and this clock is stopped it is not possible to switch to another clock. The user must switch to another clock before stopping the external clock.  
000: PLLSAI1CLK clock is selected as SAI2 clock  
001: PLLSAI2CLK clock is selected as SAI2 clock  
010: PLLSAI3CLK clock is selected as SAI2 clock  
011: External clock SAI2\_EXTCLK clock selected as SAI2 clock  
100: HSI clock selected as SAI2 clock  
Other configuration are reserved



- Bits 7:5 **SAI1SEL**: SAI1 clock source selection  
 Set and reset by software.  
 If the selected clock is the external clock and this clock is stopped it is not possible to switch to another clock.  
 The user must switch to another clock before stopping the external clock.  
 000: PLLSAI1CLK clock is selected as SAI1 clock  
 001: PLLSAI2CLK clock is selected as SAI1 clock  
 010: PLLSAI3CLK clock is selected as SAI11 clock  
 011: External clock SAI1\_EXTCLK is selected as SAI1 clock  
 100: HSI clock selected as SAI2 clock  
 Other configuration are reserved
- Bits 4:3 **ADFSDMSEL**: Digital filter for sigma delta modulator audio clock source selection  
 Set and reset by software.  
 00: SAI1clock selected as DFSDM audio clock  
 01: HSI clock selected as DFSDM audio clock  
 10: MSI clock selected as DFSDM audio clock  
 11: reserved
- Bit 2 **DFSDMSEL**: Digital filter for sigma delta modulator kernel clock source selection  
 Set and reset by software.  
 0: APB2 clock (PCLK2) selected as DFSDM kernel clock  
 1: System clock selected as DFSDM kernel clock
- Bits 1:0 **I2C4SEL[1:0]**: I2C4 clock source selection  
 These bits are set and cleared by software to select the I2C4 clock source.  
 00: PCLK selected as I2C4 clock  
 01: System clock (SYSCLK) selected as I2C4 clock  
 10: HSI16 clock selected as I2C4 clock  
 11: reserved

### 6.4.33 RCC register map

The following table gives the RCC register map and the reset values.

**Table 35. RCC register map and reset values**

Off-set	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RCC_CR	Res.	Res.	PLLSAI2RDY	PLLSAI2ON	PLLSAI1RDY	PLLSAI1ON	PLLRDY	PLLON	Res.	Res.	Res.	Res.	CSSON	HSEBYP	HSERDY	HSEON	Res.	Res.	Res.	Res.	HSIASFS	HSIRDY	HSIKERON	HSION	MSIRANG [3:0]			MSIRGSEL	MSIPLLEN	MSIRDY	MSION	
	Reset value			0	0	0	0	0	0					0	0	0	0					0	0	0	0	0	0	1	1	0	0	0	1
0x04	RCC_ICSCR	Res.	HSITRIM[6:0]						HSICAL[7:0]						MSITRIM[7:0]						MSICAL[7:0]												
	Reset value		1	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0	0	0	x	x	x	x	x	x
0x08	RCC_CFGR	Res.	MCO PRE [2:0]		MCOSEL [3:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOPWUCK	Res.	PPRE2 [2:0]		PPRE1 [2:0]		HPRE[3:0]			SWS [1:0]		SW [1:0]				
	Reset value																																



Table 35. RCC register map and reset values (continued)

Off-set	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
	Reset value		0	0	0	0	0	0	0									0		0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x0C	RCC_PLLCFGR	PLLPDIV[4:0]					PLL R [1:0]		PLLREN	Res.		PLL Q [1:0]		PLLQEN	Res.	Res.		PLLP		Res.	PLL N [6:0]							PLL M [3:0]				Res.		PLL SRC [1:0]			
	Reset value	0	0	0	0	0	0	0	0			0	0	0			0	0		0	0	1	0	0	0	0	0	0	0	0	0	0	0	0			
0x10	RCC_PLLSAI1CFGR	PLLSAI1PDIV [4:0]					PLL SAI1 R [1:0]		PLLSAI1REN	Res.		PLL SAI1 Q [1:0]		PLLSAI1QEN	Res.	Res.		PLLSAI1P		Res.	PLLSAI1N [6:0]							PLLSAI1M [3:0]				Res.		Res.			
	Reset value	0	0	0	0	0	0	0	0			0	0	0			0	0		0	0	1	0	0	0	0	0	0	0	0	0	0	0	0			
0x14	RCC_PLLSAI2CFGR	PLLSAI2PDIV [4:0]					PLL SAI2 R [1:0]		PLLSAI2REN	Res.		PLL SAI2 Q [1:0]		PLLSAI2QEN	Res.	Res.		PLLSAI2P		Res.	PLLSAI2N [6:0]							PLLSAI2M [3:0]				Res.		Res.			
	Reset value	0	0	0	0	0	0	0	0			0	0	0			0	0		0	0	1	0	0	0	0	0	0	0	0	0	0	0	0			
0x18	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48RDYIE	LSECSSIE	Res.	PLLSAI2RDYIE	PLLSAI1RDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	MSIRDYIE	LSIRDYIE	
	Reset value																										0	0	0	0	0	0	0	0	0		
0x1C	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48RDYF	LSECSSF	CSSF	PLLSAI2RDYF	PLLSAI1RDYF	PLLRDYF	HSERDYF	HSIRDYF	MSIRDYF	LSIRDYF	
	Reset value																										0	0	0	0	0	0	0	0	0	0	0
0x20	RCC_CICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48RDYC	LSECSSC	CSSC	PLLSAI2RDYC	PLLSAI1RDYC	PLLRDYC	HSERDYC	HSIRDYC	MSIRDYC	LSIRDYC	
	Reset value																										0	0	0	0	0	0	0	0	0	0	0
0x28	RCC_AHB1RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																				

Table 35. RCC register map and reset values (continued)

Off-set	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0x2C	RCC_AHB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1RST	Res.	OSPIMRST	Res.	RNGRST	HASHRST	AESRST	Res.	DCMIRST	ADCRST	OTGFSRST	Res.	Res.	Res.	GPIORST	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST		
	Reset value										0		0		0	0	0		0	0	0				0	0	0	0	0	0	0	0	0		
0x30	RCC_AHB3RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPI2RST	OSPI1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMC	RST	
	Reset value																							0	0								0		
0x38	RCC_APB1RSTR1	LPTIM1RST	OPAMP1RST	DAC1RST	PWR1RST	Res.	Res.	CAN1RST	CRSRST	I2C3RST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Res.	SPI3RST	SPI2RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST	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	
0x3C	RCC_APB1RSTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2RST	Res.	Res.	Res.	Res.	I2C4RST	LPUART1RST		
	Reset value																										0					0	0	0	
0x40	RCC_APB2RSTR	Res.	Res.	Res.	Res.	DSIRST	LTDCRST	Res.	DFSDM1RST	Res.	SAI2RST	SAI1RST	Res.	Res.	TIM17RST	TIM16RST	TIM15RST	Res.	USART1RST	TIM8RST	SPI1RST	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFG	RST
	Reset value					0	0		0		0	0			0	0	0		0	0	0	0											0	0	0
0x48	RCC_AHB1ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GFXMMUEN	DMA2DEN	TSCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLASHEN	
	Reset value														0	0	0																	1	
0x4C	RCC_AHB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1EN	Res.	OSPI1EN	Res.	RNGEN	HASHEN	AES1EN	Res.	DCMIEN	ADCEN	OTGFSEN	Res.	Res.	Res.	Res.	GPIO1EN	GPIO2EN	GPIO3EN	GPIO4EN	GPIO5EN	GPIO6EN	GPIO7EN	GPIO8EN	GPIO9EN	GPIO10EN
	Reset value										0		0		0	0	0		0	0	0					0	0	0	0	0	0	0	0	0	0
0x50	RCC_AHB3ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPI2EN	OSPI1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																							0	0										0



Table 35. RCC register map and reset values (continued)

Off-set	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x58	RCC_APB1ENR1	LPTIM1EN	OPAMPEN	DAC1EN	PWREN	Res.	Res.	CAN1EN	CRSEN	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.	SP3EN	SPI2EN	Res.	Res.	Res.	WWDGEN	RTCAPBEN	Res.	Res.	Res.	Res.	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0		0	0				0	1					0	0	0	0	0	0
0x5C	RCC_APB1ENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2EN	Res.	Res.	Res.	Res.	Res.	
	Reset value																											0						
0x60	RCC_APB2ENR	Res.	Res.	Res.	Res.	DSIEN	LTDCCEN	Res.	DFSDM1EN	Res.	SAI2EN	SAI1EN	Res.	Res.	TIM17EN	TIM16EN	TIM15EN	Res.	USART1EN	Res.	TIM8EN	SPI1EN	TIM1EN	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
0x68	RCC_AHB1SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GFXMMUSMEN	DMA2DSMEN	TSCSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value														1	1	1																	
0x6C	RCC_AHB2SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1SMEN	Res.	OSPMSMEN	Res.	RNGSMEN	HASHSMEN	AESSMEN	Res.	DCMISMEN	Res.	ADCFSSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value										1		1		1	1	1		1		1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x70	RCC_AHB3SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																	
0x78	RCC_APB1SMENR1	LPTIM1SMEN	OPAMP1SMEN	DAC1SMEN	PWRSMEN	Res.	Res.	CAN1SMEN	CRSSMEN	I2C3SMEN	I2C2SMEN	I2C1SMEN	UART5SMEN	UART4SMEN	USART3SMEN	USART2SMEN	Res.	SP3SMEN	SPI2SMEN	Res.	Res.	Res.	WWDGSMEN	RTCAPBSMEN	Res.	Res.	Res.	Res.	TIM7SMEN	TIM6SMEN	TIM5SMEN	TIM4SMEN	TIM3SMEN	TIM2SMEN
	Reset value	1	1	1	1			1	1	1	1	1	1	1	1	1		1	1				1	1					1	1	1	1	1	1



Table 35. RCC register map and reset values (continued)

Off-set	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x7C	RCC_APB1SMENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	LPTIM2SMEN					
0x80	RCC_APB2SMENR	Res.	Res.	Res.	Res.	DSISMEN	LTDCSMEN	Res.	DFSDM1SMEN	Res.	SAI2SMEN	SAI1SMEN	Res.	Res.	TIM17SMEN	TIM16SMEN	TIM15SMEN	Res.	USART1SMEN	TIM8SMEN	SPI1SMEN	TIM1SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					1	1		1		1	1			1	1	1		1	1	1	1												1
0x88	RCC_CCIPR	Res.	Res.	ADCSEL	CLK48SEL	Res.	Res.	Res.	Res.	Res.	LPTIM2SEL	LPTIM1SEL	I2C3SEL	I2C2SEL	I2C1SEL	LPUART1SEL	UART5SEL	UART4SEL	USART3SEL	USART2SEL	USART1SEL	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
0x90	RCC_BDCR	Res.	Res.	Res.	Res.	Res.	Res.	LSCOSEL	LSCOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST	RTCEN	Res.	Res.	Res.	Res.	Res.	Res.	RTCSEL [1:0]	Res.	LSECSSD	LSECSSON	LSEDRV [1:0]	LSEBYP	LSERDY	LSEON	Res.	
	Reset value							0	0								0	0							0	0	0	0	0	0	0	0	0	0
0x94	RCC_CSR	LPWRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	BORRSTF	PINRSTF	OBLRSTF	FIREWALLRSTF	RMVF	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																								
0x98	RCC_CRRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																	
0x9C	RCC_CCIPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPISEL [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.
	Reset value										0	0																						



## 7 Clock recovery system (CRS)

### 7.1 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 precise 1-ms intervals.

The synchronization signal can also be derived from the LSE oscillator output or it can be generated by user software.

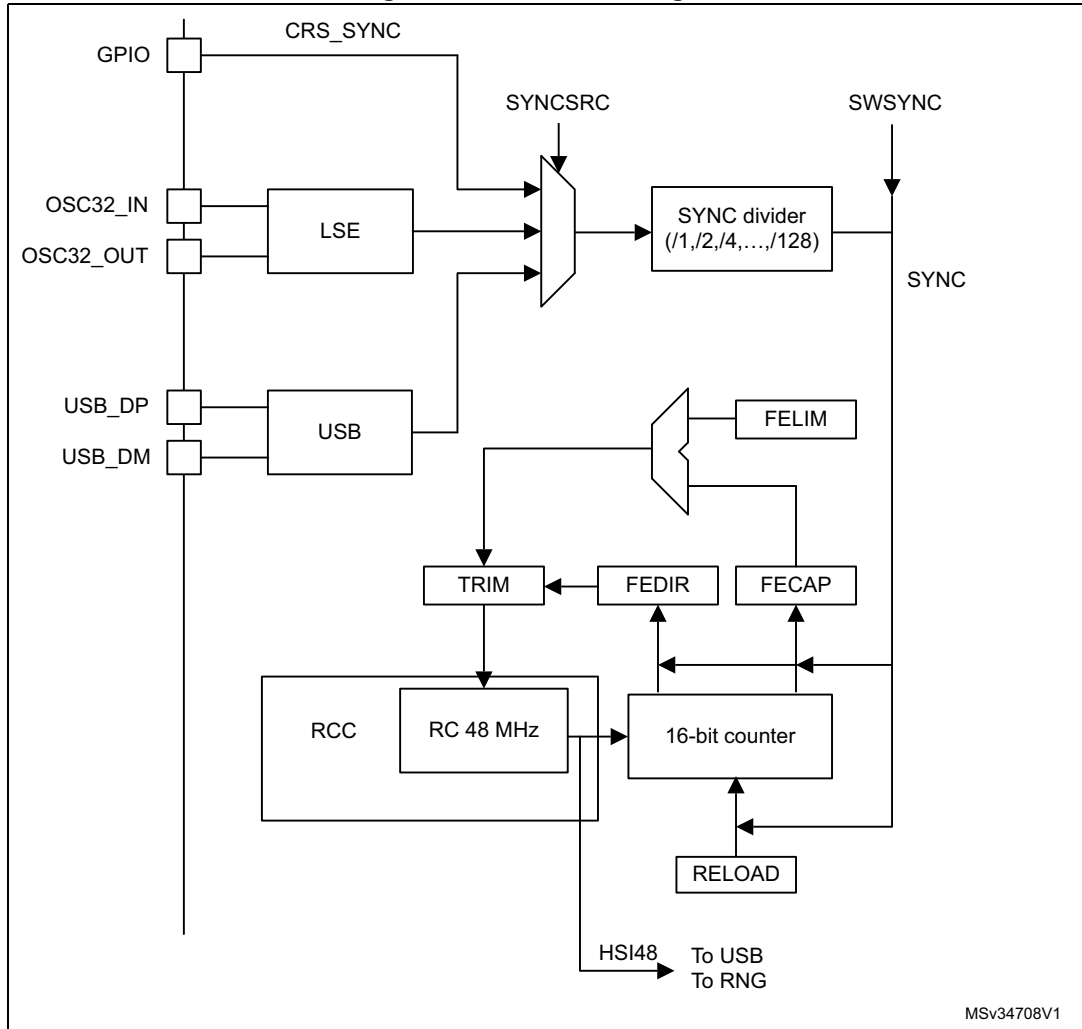
### 7.2 CRS main features

- Selectable synchronization source with programmable prescaler and polarity:
  - External pin
  - LSE oscillator output
  - USB SOF 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)

## 7.3 CRS functional description

### 7.3.1 CRS block diagram

Figure 20. CRS block diagram



### 7.3.2 Synchronization input

The CRS synchronization (SYNC) source, selectable through the CRS\_CFGR register, can be the signal from the LSE clock or the USB SOF signal. For a better robustness of the SYNC input, a simple digital filter (2 out of 3 majority votes, sampled by the RC48 clock) is implemented to filter out any glitches. This source signal also has a configurable polarity and can then be divided by a programmable binary prescaler to obtain a synchronization signal in a suitable frequency range (usually around 1 kHz).

For more information on the CRS synchronization source configuration, refer to [Section 7.6.2: CRS configuration register \(CRS\\_CFGR\)](#).

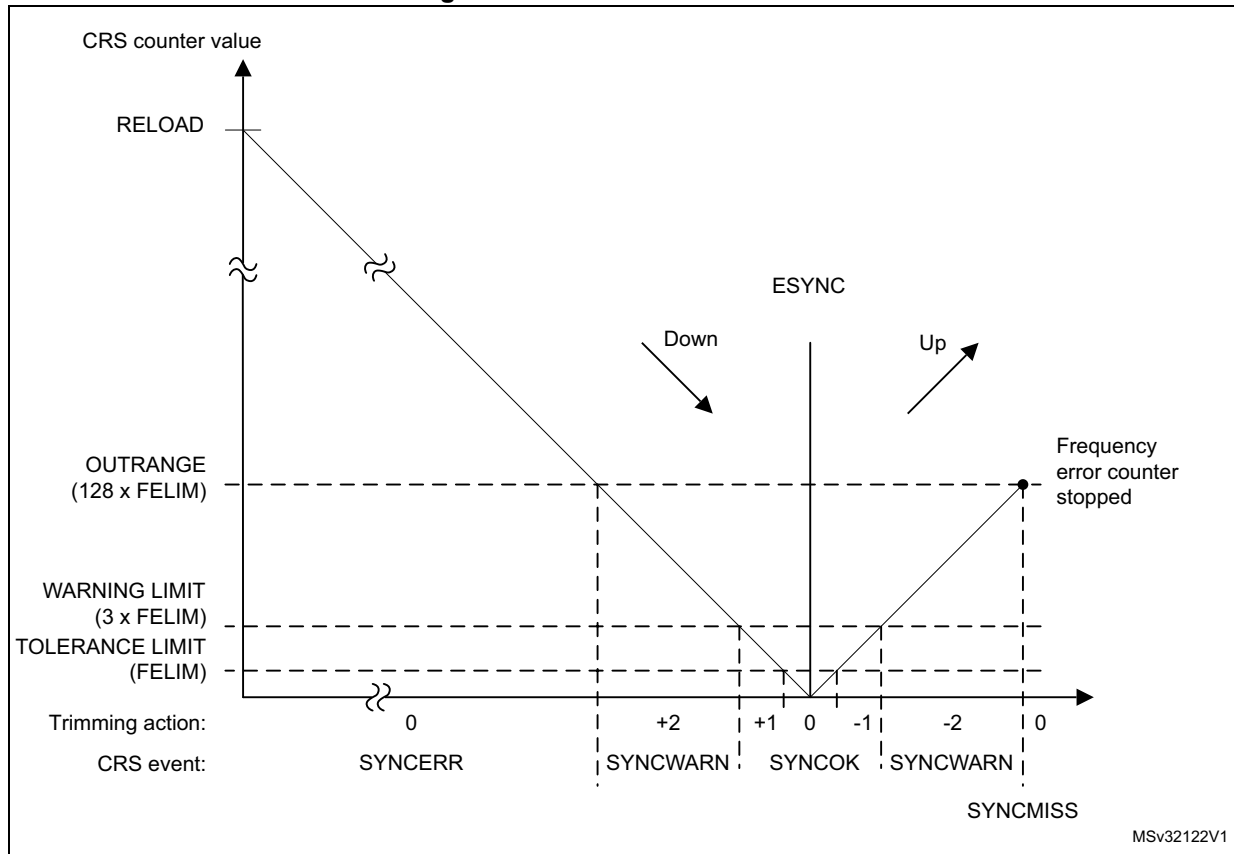
It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS\_CR register.

### 7.3.3 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 should be incremented), while when it is detected during the upcounting phase it means that the actual frequency is higher (and that the TRIM value should be decremented).

Figure 21. CRS counter behavior





### 7.3.4 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 \* FELIM value
- OUTRANGE (error limit), defined as 128 \* 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 and that then, no trimming action is necessary.
  - 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 will not be 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.*

### 7.3.5 CRS initialization and configuration

#### RELOAD value

The RELOAD value should 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 in order 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:

$$FELIM = (f_{TARGET} / f_{SYNC}) * STEP[\%] / 100\% / 2$$

The result should be always rounded up to the nearest integer value in order to obtain the best trimming response. If frequent trimming actions are not wanted in the application, the trimming hysteresis can be increased by increasing slightly the FELIM value.

The reset value of the FELIM field corresponds to  $(f_{TARGET} / f_{SYNC}) = 48000$  and to a typical trimming step size of 0.14%.

**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).

**7.4 CRS low-power modes**

**Table 36. 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.
Standby	The CRS stops operating until the Stop or Standby mode is exited and the HSI48 oscillator restarted.

**7.5 CRS interrupts**

**Table 37. Interrupt control bits**

Interrupt event	Event flag	Enable control bit	Clear flag bit
Expected synchronization	ESYNCF	ESYNCE	ESYNCC
Synchronization OK	SYNCOKF	SYNCOKE	SYNCOKC
Synchronization warning	SYNCWARNF	SYNCWARNIE	SYNCWARNC
Synchronization or trimming error (TRIMOVF, SYNCMISS, SYNCERR)	ERRF	ERRIE	ERRC

## 7.6 CRS registers

Refer to [Section 1.2 on page 75](#) of the reference manual for a list of abbreviations used in register descriptions.

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

### 7.6.1 CRS control register (CRS\_CR)

Address offset: 0x00

Reset value: 0x0000 4000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TRIM[6:0]							SW SYNC	AUTO TRIMEN	CEN	Res.	ESYNC IE	ERRIE	SYNC WARNIE	SYNC OKIE
	rw	rw	rw	rw	rw	rw	rw	rt_w	rw	rw		rw	rw	rw	rw

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

Bits 14:8 **TRIM[6:0]**: HSI48 oscillator smooth trimming

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 64, 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 7.3.4: Frequency error evaluation and automatic trimming](#) for more details.

0: Automatic trimming disabled, TRIM bits can be adjusted by the user.

1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.

Bit 5 **CEN**: Frequency error counter enable

This bit enables the oscillator clock for the frequency error counter.

0: Frequency error counter disabled

1: Frequency error counter enabled

When this bit is set, the CRS\_CFGR register is write-protected and cannot be modified.

Bit 4 Reserved, must be kept at reset value.

- Bit 3 **ESYNCE**: 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

### 7.6.2 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]							
r/w		r/w	r/w		r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RELOAD[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

- Bit 31 **SYNCPOL**: SYNC polarity selection
  - This bit is set and cleared by software to select the input polarity for the SYNC signal source.
  - 0: SYNC active on rising edge (default)
  - 1: SYNC active on falling edge

Bit 30 Reserved, must be kept at reset value.

- Bits 29:28 **SYNCSRC[1:0]**: SYNC signal source selection
  - These bits are set and cleared by software to select the SYNC signal source.
  - 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 will not be generated by the host. No SYNC signal will therefore be 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 7.3.4: Frequency error evaluation and automatic trimming](#) 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 7.3.3: Frequency error measurement](#) for more details about counter behavior.

### 7.6.3 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 7.3.4: Frequency error evaluation and automatic trimming](#) 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 signaled
- 1: Trimming error signaled

**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 should be taken. At this point, the frequency error counter is stopped (waiting for a next SYNC) and an interrupt is generated if the ERRIE bit is set in the CRS\_CR register. It is cleared by software by setting the ERRC bit in the CRS\_ICR register.

0: No SYNC missed error signaled  
1: SYNC missed error signaled

**Bit 8 SYNCERR:** SYNC error

This flag is set by hardware when the SYNC pulse arrives before the ESYNC event and the measured frequency error is greater than or equal to  $FELIM * 128$ . This means that the frequency error is too big (internal frequency too low) to be compensated by adjusting the TRIM value, and that some other action should be taken. An interrupt is generated if the ERRIE bit is set in the CRS\_CR register. It is cleared by software by setting the ERRC bit in the CRS\_ICR register.

0: No SYNC error signaled  
1: SYNC error signaled

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

**Bit 3 ESYNCF:** Expected SYNC flag

This flag is set by hardware when the frequency error counter reached a zero value. An interrupt is generated if the ESYNCIE bit is set in the CRS\_CR register. It is cleared by software by setting the ESYNCC bit in the CRS\_ICR register.

0: No expected SYNC signaled  
1: Expected SYNC signaled

**Bit 2 ERRF:** Error flag

This flag is set by hardware in case of any synchronization or trimming error. It is the logical OR of the TRIMOVF, SYNCMISS and SYNCERR bits. An interrupt is generated if the ERRIE bit is set in the CRS\_CR register. It is cleared by software in reaction to setting the ERRC bit in the CRS\_ICR register, which clears the TRIMOVF, SYNCMISS and SYNCERR bits.

0: No synchronization or trimming error signaled  
1: Synchronization or trimming error signaled

**Bit 1 SYNCWARNF:** SYNC warning flag

This flag is set by hardware when the measured frequency error is greater than or equal to  $FELIM * 3$ , but smaller than  $FELIM * 128$ . This means that to compensate the frequency error, the TRIM value must be adjusted by two steps or more. An interrupt is generated if the SYNCWARNIE bit is set in the CRS\_CR register. It is cleared by software by setting the SYNCWARNC bit in the CRS\_ICR register.

0: No SYNC warning signaled  
1: SYNC warning signaled

**Bit 0 SYNCOKF:** SYNC event OK flag

This flag is set by hardware when the measured frequency error is smaller than  $FELIM * 3$ . This means that either no adjustment of the TRIM value is needed or that an adjustment by one trimming step is enough to compensate the frequency error. An interrupt is generated if the SYNCOKIE bit is set in the CRS\_CR register. It is cleared by software by setting the SYNCOKC bit in the CRS\_ICR register.

0: No SYNC event OK signaled  
1: SYNC event OK signaled

### 7.6.4 CRS interrupt flag clear register (CRS\_ICR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

7.6.5 CRS register map

Table 38. CRS 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	CRS_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[6:0]						SWSYNC	AUTOTRIMEN	CEN	Res.	ESYNCE	ERRIE	SYNCWARNIE	SYNCKOIE	
	Reset value																		1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	CRS_CFGR	SYNCPOL	Res.	SYNC SRC [1:0]		Res.	SYNC DIV [2:0]		FELIM[7:0]							RELOAD[15:0]																	
	Reset value	0		1	0		0	0	0	0	0	0	1	0	0	0	1	0	1	0	1	1	1	0	1	1	0	1	1	1	1	1	1
0x08	CRS_ISR	FECAP[15:0]															FEDIR	Res.	Res.	Res.	Res.	Res.	TRIMOVF	SYNCMISS	SYNCERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNCWARNF	SYNCKOIF
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												0	0	0
0x0C	CRS_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																													0	0	0	0

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.





## 8 General-purpose I/Os (GPIO)

### 8.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx\_MODER, GPIOx\_OTYPER, GPIOx\_OSPEEDR and GPIOx\_PUPDR), two 32-bit data registers (GPIOx\_IDR and GPIOx\_ODR) and a 32-bit set/reset register (GPIOx\_BSRR). In addition all GPIOs have a 32-bit locking register (GPIOx\_LCKR) and two 32-bit alternate function selection registers (GPIOx\_AFRH and GPIOx\_AFLR).

### 8.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx\_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx\_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx\_BSRR) for bitwise write access to GPIOx\_ODR
- Locking mechanism (GPIOx\_LCKR) provided to freeze the 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

### 8.3 GPIO functional description

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

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

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

Figure 22 and Figure 23 show the basic structures of a standard and a 5-Volt tolerant I/O port bit, respectively. Table 39 gives the possible port bit configurations.

Figure 22. Basic structure of an I/O port bit

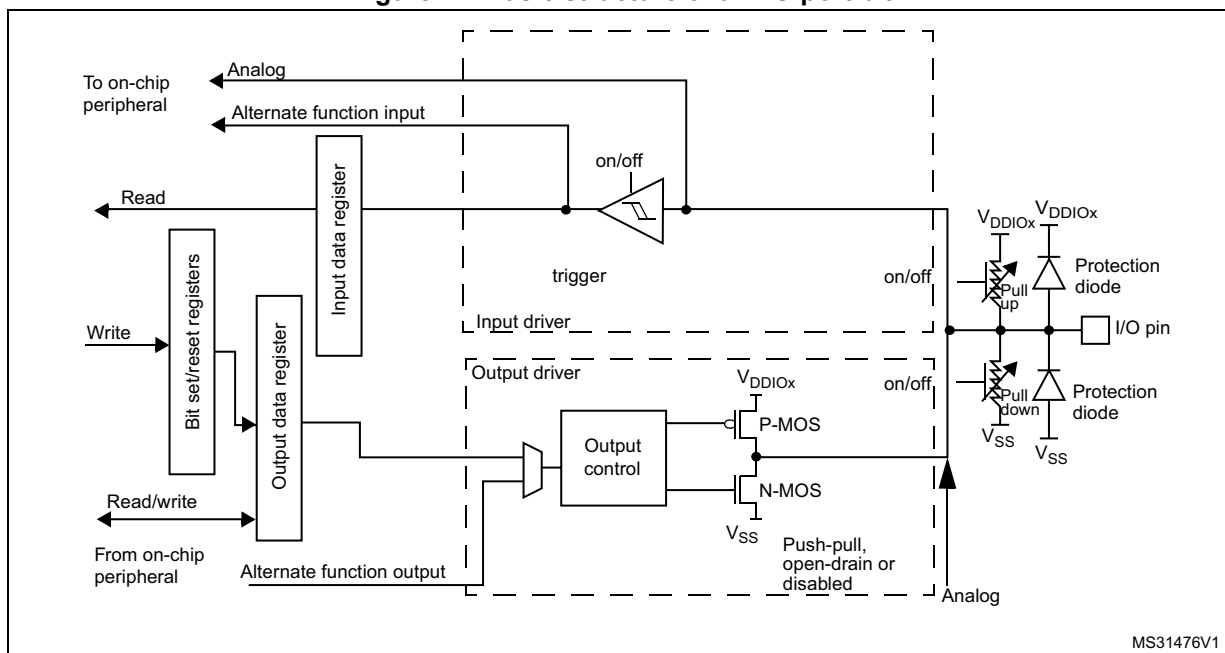
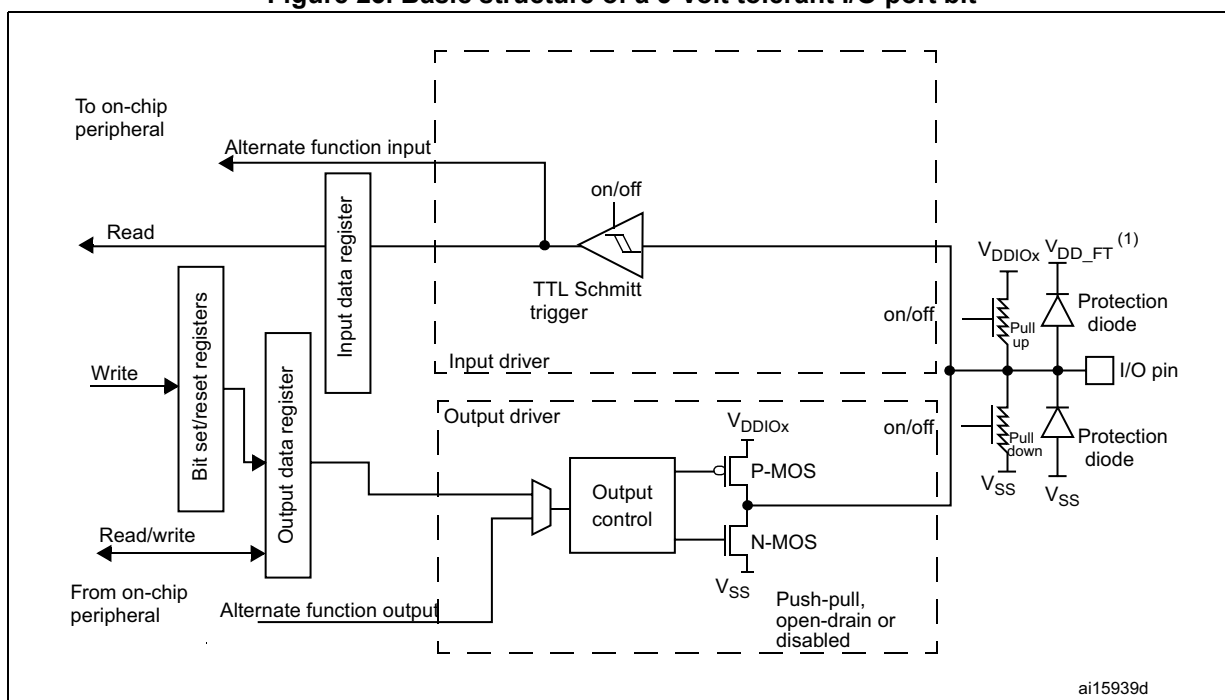


Figure 23. Basic structure of a 5-Volt tolerant I/O port bit



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

Table 39. 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.

### 8.3.1 General-purpose I/O (GPIO)

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

The debug pins are in AF pull-up/pull-down after reset:

- PA15: JTDI in pull-up
- PA14: JTCK/SWCLK in pull-down
- PA13: JTMS/SWDAT in pull-up
- PB4: NJTRST in pull-up
- PB3: JTDO in floating state no pull-up/pull-down

PH3/BOOT0 is in input mode during the reset until at least the end of the option byte loading phase. See [Section 8.3.15: 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.

### 8.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there can be no conflict between peripherals available on the same I/O pin.

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

- After reset the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx\_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, 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, DAC, OPAMP, and COMP, configure the desired I/O in analog mode in the GPIOx\_MODER register and configure the required function in the ADC, DAC, OPAMP, 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.

### 8.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx\_MODER, GPIOx\_OTYPER, GPIOx\_OSPEEDR, GPIOx\_PUPDR) to configure up to 16 I/Os. The GPIOx\_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx\_OTYPER and GPIOx\_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIOx\_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

### 8.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIOx\_IDR and GPIOx\_ODR). GPIOx\_ODR stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (GPIOx\_IDR), a read-only register.

See [Section 8.4.5: GPIO port input data register \(GPIOx\\_IDR\) \(x = A to I\)](#) and [Section 8.4.6: GPIO port output data register \(GPIOx\\_ODR\) \(x = A to I\)](#) for the register descriptions.

### 8.3.5 I/O data bitwise handling

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

To each bit in GPIOx\_ODR, correspond two control bits in GPIOx\_BSRR: BS(i) and BR(i). When written to 1, bit BS(i) **sets** the corresponding ODR(i) bit. When written to 1, bit BR(i) **resets** the ODR(i) corresponding bit.

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

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

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

### 8.3.6 GPIO locking mechanism

It is possible to freeze the GPIO control registers by applying a specific write sequence to the GPIOx\_LCKR register. The frozen registers are GPIOx\_MODER, GPIOx\_OTYPER, GPIOx\_OSPEEDR, GPIOx\_PUPDR, GPIOx\_AFRL and GPIOx\_AFRH.

To write the GPIOx\_LCKR register, a specific write / read sequence has to be applied. When the right LOCK sequence is applied to bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence has been applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx\_LCKR bit freezes the corresponding bit in the control registers (GPIOx\_MODER, GPIOx\_OTYPER, GPIOx\_OSPEEDR, GPIOx\_PUPDR, GPIOx\_AFRL and GPIOx\_AFRH).

The LOCK sequence (refer to [Section 8.4.8: GPIO port configuration lock register \(GPIOx\\_LCKR\) \(x = A to I\)](#)) can only be performed using a word (32-bit long) access to the GPIOx\_LCKR register due to the fact that GPIOx\_LCKR bit 16 has to be set at the same time as the [15:0] bits.

For more details refer to LCKR register description in [Section 8.4.8: GPIO port configuration lock register \(GPIOx\\_LCKR\) \(x = A to I\)](#).

### 8.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, the user can connect an alternate function to some other pin as required by the application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using the GPIOx\_AFRL and GPIOx\_AFRH alternate function registers. The application can thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the device datasheet.

### 8.3.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port can be configured in input, output or alternate function mode (the port must not be configured in analog mode). Refer to [Section 16: Extended interrupts and events controller \(EXTI\)](#) [Section 16.3.2: Wakeup event management](#).

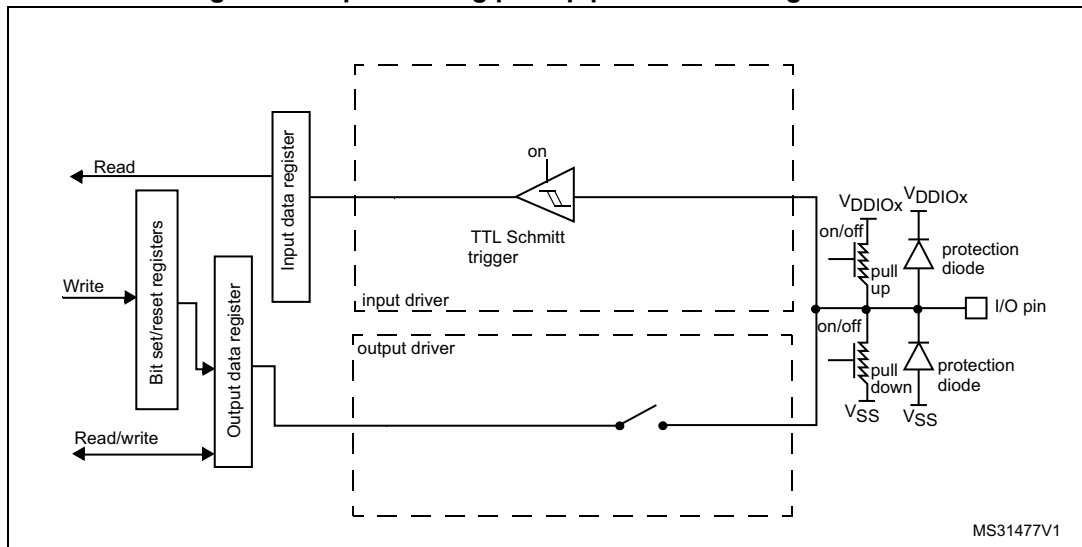
### 8.3.9 Input configuration

When the I/O port is programmed as input:

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

Figure 24 shows the input configuration of the I/O port bit.

**Figure 24. Input floating/pull up/pull down configurations**



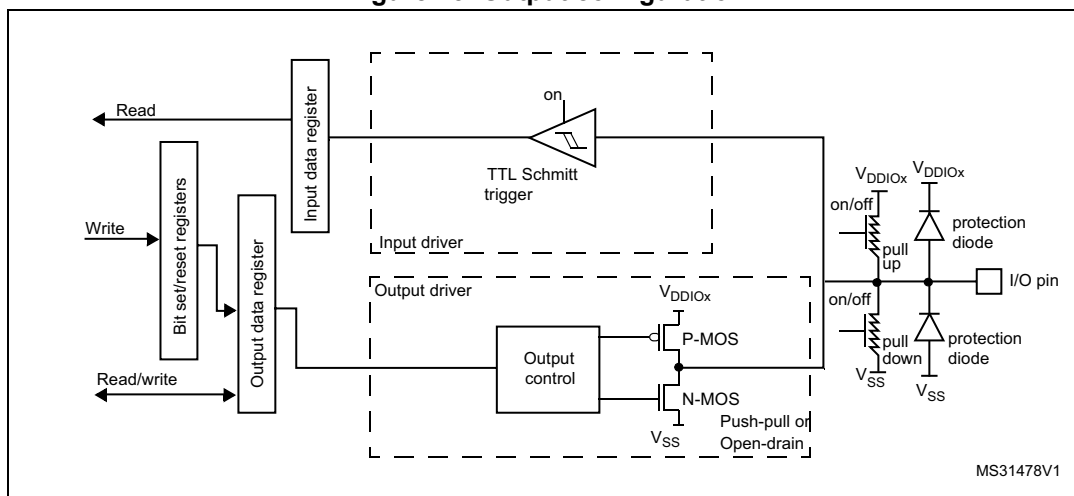
### 8.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open drain mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register leaves the port in Hi-Z (the P-MOS is never activated)
  - Push-pull mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register activates the P-MOS
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx\_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state
- A read access to the output data register gets the last written value

Figure 25 shows the output configuration of the I/O port bit.

Figure 25. Output configuration



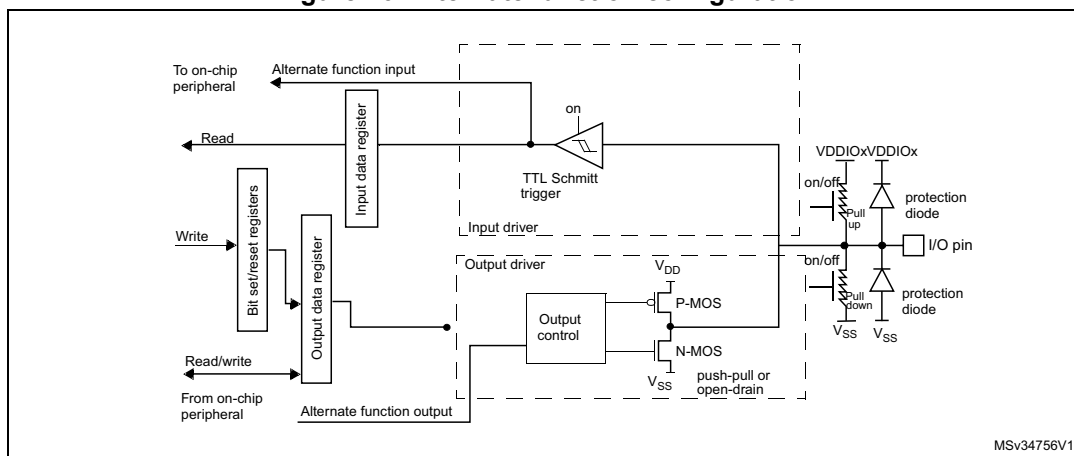
### 8.3.11 Alternate function configuration

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

- The output buffer can be configured in open-drain or push-pull mode
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data)
- The Schmitt trigger input is activated
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx\_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state

Figure 26 shows the Alternate function configuration of the I/O port bit.

Figure 26. Alternate function configuration





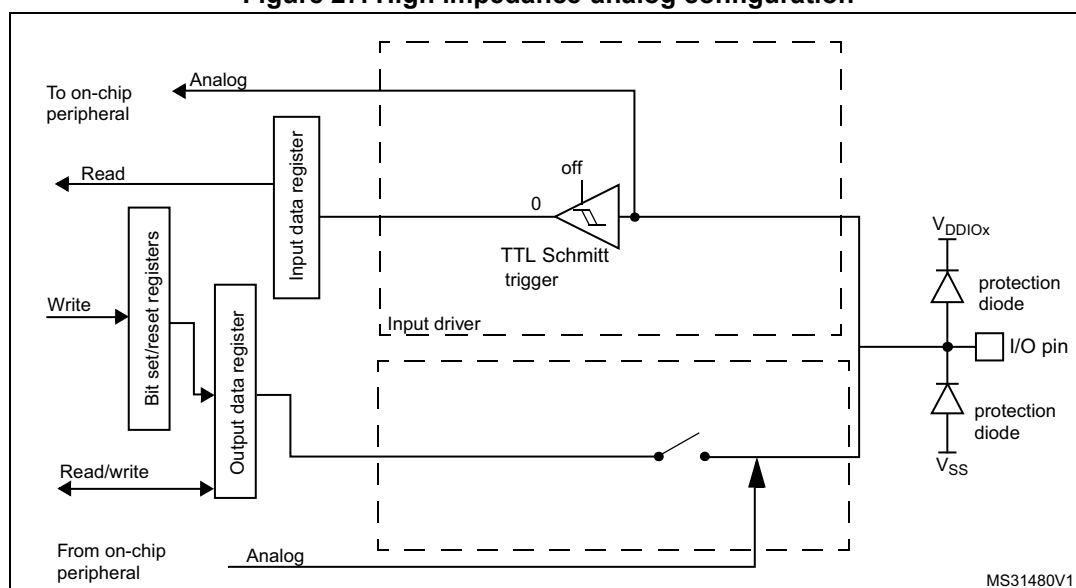
### 8.3.12 Analog configuration

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

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

Figure 27 shows the high-impedance, analog-input configuration of the I/O port bits.

Figure 27. High impedance-analog configuration



### 8.3.13 Using the HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched OFF (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched ON (by setting the HSEON or LSEON bit in the RCC\_CSR register) the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect.

When the oscillator is configured in a user external clock mode, only the pin is reserved for clock input and the OSC\_OUT or OSC32\_OUT pin can still be used as normal GPIO.

### 8.3.14 Using the GPIO pins in the RTC supply domain

The PC13/PC14/PC15 GPIO functionality is lost when the core supply domain is powered off (when the device enters Standby mode). In this case, if their GPIO configuration is not bypassed by the RTC configuration, these pins are set in an analog input mode.

For details about I/O control by the RTC, refer to [Section 42.3: RTC functional description](#).

### 8.3.15 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.

## 8.4 GPIO registers

This section gives a detailed description of the GPIO registers.

For a summary of register bits, register address offsets and reset values, refer to [Table 40](#).

The peripheral registers can be written in word, half word or byte mode.

### 8.4.1 GPIO port mode register (GPIOx\_MODER) (x =A to I)

Address offset:0x00

Reset value:

- 0xABFF FFFF (for port A)
- 0xFFFF FEBF (for port B)
- 0xFFFF FFFF (for ports C..G), I
- 0x0000 000F (for port H)

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]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **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)

### 8.4.2 GPIO port output type register (GPIOx\_OTYPER) (x = A to I)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 15:0 **OT[15:0]**: Port x configuration I/O pin y (y = 15 to 0)

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

- 0: Output push-pull (reset state)
- 1: Output open-drain

### 8.4.3 GPIO port output speed register (GPIOx\_OSPEEDR) (x = A to I)

Address offset: 0x08

Reset value: 0x0C00 0000 (for port A)

Reset value: 0x0000 0000 (for the other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15 [1:0]		OSPEED14 [1:0]		OSPEED13 [1:0]		OSPEED12 [1:0]		OSPEED11 [1:0]		OSPEED10 [1:0]		OSPEED9 [1:0]		OSPEED8 [1:0]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED7 [1:0]		OSPEED6 [1:0]		OSPEED5 [1:0]		OSPEED4 [1:0]		OSPEED3 [1:0]		OSPEED2 [1:0]		OSPEED1 [1:0]		OSPEED0 [1:0]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **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: High speed
- 11: Very high speed

*Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed..*

### 8.4.4 GPIO port pull-up/pull-down register (GPIOx\_PUPDR) (x = A to I)

Address offset: 0x0C

Reset value: 0x6400 0000 (for port A)



Reset value: 0x0000 0100 (for port B)

Reset value: 0x0000 0000 (for other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
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
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	rw	rw	rw	rw	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

### 8.4.5 GPIO port input data register (GPIOx\_IDR) (x = A to I)

Address offset: 0x10

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 15:0 **ID[15:0]**: Port x input data I/O pin y (y = 15 to 0)

These bits are read-only. They contain the input value of the corresponding I/O port.

### 8.4.6 GPIO port output data register (GPIOx\_ODR) (x = A to I)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
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 **OD[15:0]**: Port output data I/O pin y (y = 15 to 0)

These bits can be read and written by software.

*Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx\_BSRR or GPIOx\_BRR registers (x = A..F).*

### 8.4.7 GPIO port bit set/reset register (GPIOx\_BSRR) (x = A to I)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **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

### 8.4.8 GPIO port configuration lock register (GPIOx\_LCKR) (x = A to I)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

*Note: A specific write sequence is used to write to the GPIOx\_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.*

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

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



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 16 **LCKK**: Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx\_LCKR register is locked until the next MCU reset or peripheral reset.

LOCK key write sequence:

WR LCKR[16] = '1' + LCKR[15:0]

WR LCKR[16] = '0' + LCKR[15:0]

WR LCKR[16] = '1' + LCKR[15:0]

RD LCKR

RD LCKR[16] = '1' (this read operation is optional but it confirms that the lock is active)

*Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.*

*Any error in the lock sequence aborts the lock.*

*After the first lock sequence on any bit of the port, any read access on the LCKK bit will return '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

### 8.4.9 GPIO alternate function low register (GPIOx\_AFRL) (x = A to I)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	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
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **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

### 8.4.10 GPIO alternate function high register (GPIOx\_AFRH) (x = A to I)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **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

**8.4.11 GPIO port bit reset register (GPIOx\_BRR) (x = A to I)**

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

Bits 15:0 **BR[15:0]**: Port x reset IO pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Reset the corresponding ODx bit



### 8.4.12 GPIO register map

The following table gives the GPIO register map and reset values.

**Table 40. GPIO register map and reset values**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOA_MODER	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	1 0		1 0		1 0		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1	
0x00	GPIOB_MODER	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 0		1 0		1 1		1 1		1 1	
0x00	GPIOx_MODER (where x = C..I)	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1		1 1	
0x04	GPIOx_OTYPER (where x = A..I)	Res. Res.		Res. Res.		Res. Res.		Res. Res.		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	
0x08	GPIOA_OSPEEDR	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0 0		0 0		1 1		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0	
0x08	GPIOx_OSPEEDR (where x = B..H)	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0	
0x0C	GPIOA_PUPDR	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value	0 1		1 0		0 1		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0	
0x0C	GPIOB_PUPDR	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value	0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 1		0 0		0 0		0 0		0 0	
0x0C	GPIOx_PUPDR (where x = C..I)	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value	0 0		0 0		0 0		0 0		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	GPIOx_IDR (where x = A..I)	Res. Res.		Res. Res.		Res. Res.		Res. Res.		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	



Table 40. GPIO register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x14	GPIOx_ODR (where x = A..I)	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	GPIOx_BSRR (where x = A..I)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	GPIOx_LCKR (where x = A..I)	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	GPIOx_AFRL (where x = A..I)	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	GPIOx_AFRH (where x = A..I)	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]				AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	GPIOx_BRR (where x = A..I)	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

## 9 System configuration controller (SYSCFG)

### 9.1 SYSCFG main features

The STM32L4Rxxx and STM32L4Sxxx 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 write protection and software erase
- Configuring FPU interrupts
- Enabling the firewall
- Enabling /disabling I<sup>2</sup>C Fast-mode Plus driving capability on some I/Os and voltage booster for I/Os analog switches.

### 9.2 SYSCFG registers

#### 9.2.1 SYSCFG memory remap register (SYSCFG\_MEMRMP)

This register is used for specific configurations on memory remap.

Address offset: 0x00

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	Res	Res	Res	Res	Res	Res	Res	Res	Res	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	FB MODE	Res	Res	Res	Res	Res	MEM_MODE		
							rw						rw	rw	rw

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

Bit 8 **FB\_MODE**: Flash Bank mode selection

- 0: Flash Bank 1 mapped at 0x0800 0000 (and aliased @0x0000 0000<sup>(1)</sup>) and Flash Bank 2 mapped at 0x0808 0000 (and aliased at 0x0008 0000)
- 1: Flash Bank2 mapped at 0x0800 0000 (and aliased @0x0000 0000<sup>(1)</sup>) and Flash Bank 1 mapped at 0x0808 0000 (and aliased at 0x0008 0000)

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

Bits 2:0 **MEM\_MODE**: 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 pin and the option bit 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 0x00000000<sup>(1)</sup>.
  - 001: System Flash memory mapped at 0x00000000.
  - 010: FMC bank 1 (NOR/PSRAM 1 and 2) mapped at 0x00000000.
  - 011: SRAM1 mapped at 0x00000000.
  - 100: OCTOSPI1 memory mapped at 0x00000000
  - 101: OCTOSPI2 memory mapped at 0x00000000
  - 111: Reserved

1. When BFB2 bit is set, the system memory remains aliased at @0x0000 0000

**Note:** *When the FSMC is remapped at address 0x0000 0000, only the first two regions of Bank 1 memory controller (Bank1 NOR/PSRAM 1 and NOR/PSRAM 2) can be remapped. In remap mode, the CPU can access the external memory via ICode bus instead of System bus which boosts up the performance.*

### 9.2.2 SYSCFG configuration register 1 (SYSCFG\_CFGR1)

Address offset: 0x04

Reset value: 0x7C00 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FPU_IE[5..0]						Res	Res	I2C4_FMP	I2C3_FMP	I2C2_FMP	I2C1_FMP	I2C_PB9_FMP	I2C_PB8_FMP	I2C_PB7_FMP	I2C_PB6_FMP
rw	rw	rw	rw	rw	rw			rw	rw	rw	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	ANAS WVDD	BOOST EN	Res	Res	Res	Res	Res	Res	Res	FWDIS
						rw	rw								rc_w0

- Bits 31:26 **FPU\_IE[5..0]**: Floating Point Unit interrupts enable bits  
 FPU\_IE[5]: Inexact interrupt enable  
 FPU\_IE[4]: Input denormal interrupt enable  
 FPU\_IE[3]: Overflow interrupt enable  
 FPU\_IE[2]: underflow interrupt enable  
 FPU\_IE[1]: Divide-by-zero interrupt enable  
 FPU\_IE[0]: Invalid operation interrupt enable
- Bits 25:24 Reserved, must be kept at reset value.
- Bit 23 **I2C4\_FMP**: Fast-mode Plus driving capability activation  
 This bit enables the Fm+ driving mode on I2C4 pins selected through AF selection bits.  
 0: Fm+ mode is not enabled on I2C4 pins selected through AF selection bits  
 1: Fm+ mode is enabled on I2C4 pins selected through AF selection bits.
- Bit 22 **I2C3\_FMP**: I2C3 Fast-mode Plus driving capability activation  
 This bit enables the Fm+ driving mode on I2C3 pins selected through AF selection bits.  
 0: Fm+ mode is not enabled on I2C3 pins selected through AF selection bits  
 1: Fm+ mode is enabled on I2C3 pins selected through AF selection bits.
- Bit 21 **I2C2\_FMP**: I2C2 Fast-mode Plus driving capability activation  
 This bit enables the Fm+ driving mode on I2C2 pins selected through AF selection bits.  
 0: Fm+ mode is not enabled on I2C2 pins selected through AF selection bits  
 1: Fm+ mode is enabled on I2C2 pins selected through AF selection bits.
- Bit 20 **I2C1\_FMP**: I2C1 Fast-mode Plus driving capability activation  
 This bit enables the Fm+ driving mode on I2C1 pins selected through AF selection bits.  
 0: Fm+ mode is not enabled on I2C1 pins selected through AF selection bits  
 1: Fm+ mode is enabled on I2C1 pins selected through AF selection bits.
- Bit 19 **I2C\_PB9\_FMP**: Fast-mode Plus (Fm+) driving capability activation on PB9  
 This bit enables the Fm+ driving mode for PB9.  
 0: PB9 pin operates in standard mode.  
 1: Fm+ mode enabled on PB9 pin, and the Speed control is bypassed.
- Bit 18 **I2C\_PB8\_FMP**: Fast-mode Plus (Fm+) driving capability activation on PB8  
 This bit enables the Fm+ driving mode for PB8.  
 0: PB8 pin operates in standard mode.  
 1: Fm+ mode enabled on PB8 pin, and the Speed control is bypassed.
- Bit 17 **I2C\_PB7\_FMP**: Fast-mode Plus (Fm+) driving capability activation on PB7  
 This bit enables the Fm+ driving mode for PB7.  
 0: PB7 pin operates in standard mode.  
 1: Fm+ mode enabled on PB7 pin, and the Speed control is bypassed.
- Bit 16 **I2C\_PB6\_FMP**: Fast-mode Plus (Fm+) driving capability activation on PB6  
 This bit enables the Fm+ driving mode for PB6.  
 0: PB6 pin operates in standard mode.  
 1: Fm+ mode enabled on PB6 pin, and the Speed control is bypassed.
- Bits 15:10 Reserved, must be kept at reset value.
- Bit 9 **ANASWVDD**: GPIO analog switch control voltage selection  
 0: I/O analog switches supplied by VDDA or booster when booster is ON  
 1: I/O analog switches supplied by VDD.  
 Refer to [Table 41](#) for bit 9 setting.

Bit 8 **BOOSTEN**: I/O analog switch voltage booster enable

0: I/O analog switches are supplied by V<sub>DDA</sub> voltage. This is the recommended configuration when using the ADC in high V<sub>DDA</sub> voltage operation.

1: I/O analog switches are supplied by a dedicated voltage booster (supplied by V<sub>DD</sub>). This is the recommended configuration when using the ADC in low V<sub>DDA</sub> voltage operation.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **FWDIS**: Firewall disable

This bit is cleared by software to protect the access to the memory segments according to the Firewall configuration. Once enabled, the firewall cannot be disabled by software. Only a system reset set the bit.

0 : Firewall protection enabled

1 : Firewall protection disabled

Table 41 describes when the bit 9 (ANASWVDD) and the bit 8 (BOOSTEN) should be set or reset depending on the voltage settings.

**Table 41. BOOSTEN and ANASWVDD set/reset**

VDD	VDDA	BOOSTEN	ANASWVDD
-	> 2.4 V	0	0
> 2.4 V	< 2.4 V	0	1
< 2.4 V	< 2.4 V	1	0

### 9.2.3 SYSCFG external interrupt configuration register 1 (SYSCFG\_EXTICR1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 15:12 **EXTI3[3:0]**: EXTI 3 configuration bits

These bits are written by software to select the source input for the EXTI3 external interrupt.

0000: PA[3] pin

0001: PB[3] pin

0010: PC[3] pin

0011: PD[3] pin

0100: PE[3] pin

0101: PF[3] pin

0110: PG[3] pin

0111: PH[3] pin

1000: PI[3] pin

Bits 11:8 **EXTI2[3:0]**: EXTI 2 configuration bits

These bits are written by software to select the source input for the EXTI2 external interrupt.

0000: PA[2] pin

0001: PB[2] pin

0010: PC[2] pin

0011: PD[2] pin

0100: PE[2] pin

0101: PF[2] pin

0110: PG[2] pin

0111: PH[2] pin

1000: PI[2] pin

Bits 7:4 **EXTI1[3:0]**: EXTI 1 configuration bits

These bits are written by software to select the source input for the EXTI1 external interrupt.

0000: PA[1] pin

0001: PB[1] pin

0010: PC[1] pin

0011: PD[1] pin

0100: PE[1] pin

0101: PF[1] pin

0110: PG[1] pin

0111: PH[1] pin

1000: PI[1] pin

Bits 3:0 **EXTI0[3:0]**: EXTI 0 configuration bits

These bits are written by software to select the source input for the EXTI0 external interrupt.

0000: PA[0] pin

0001: PB[0] pin

0010: PC[0] pin

0011: PD[0] pin

0100: PE[0] pin

0101: PF[0] pin

0110: PG[0] pin

0111: PH[0] pin

1000: PI[0] pin

### 9.2.4 SYSCFG external interrupt configuration register 2 (SYSCFG\_EXTICR2)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 15:12 **EXTI7[3:0]**: EXTI 7 configuration bits

These bits are written by software to select the source input for the EXTI7 external interrupt.

- 0000: PA[7] pin
- 0001: PB[7] pin
- 0010: PC[7] pin
- 0011: PD[7] pin
- 0100: PE[7] pin
- 0101: PF[7] pin
- 0110: PG[7] pin
- 0111: PH[7] pin
- 1000: PI[7] pin



Bits 11:8 **EXTI6[3:0]**: EXTI 6 configuration bits

These bits are written by software to select the source input for the EXTI6 external interrupt.

- 0000: PA[6] pin
- 0001: PB[6] pin
- 0010: PC[6] pin
- 0011: PD[6] pin
- 0100: PE[6] pin
- 0101: PF[6] pin
- 0110: PG[6] pin
- 0111: PH[6] pin
- 1000: PI[6] pin

Bits 7:4 **EXTI5[3:0]**: EXTI 5 configuration bits

These bits are written by software to select the source input for the EXTI5 external interrupt.

- 0000: PA[5] pin
- 0001: PB[5] pin
- 0010: PC[5] pin
- 0011: PD[5] pin
- 0100: PE[5] pin
- 0101: PF[5] pin
- 0110: PG[5] pin
- 0111: PH[5] pin
- 1000: PI[5] pin

Bits 3:0 **EXTI4[3:0]**: EXTI 4 configuration bits

These bits are written by software to select the source input for the EXTI4 external interrupt.

- 0000: PA[4] pin
- 0001: PB[4] pin
- 0010: PC[4] pin
- 0011: PD[4] pin
- 0100: PE[4] pin
- 0101: PF[4] pin
- 0110: PG[4] pin
- 0111: PH[4] pin
- 1000: PI[4] pin

*Note:* Some of the I/O pins mentioned in the above register may not be available on small packages.

### 9.2.5 SYSCFG external interrupt configuration register 3 (SYSCFG\_EXTICR3)

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
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w



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

Bits 15:12 **EXTI11[3:0]**: EXTI 11 configuration bits

These bits are written by software to select the source input for the EXTI11 external interrupt.

0000: PA[11] pin  
0001: PB[11] pin  
0010: PC[11] pin  
0011: PD[11] pin  
0100: PE[11] pin  
0101: PF[11] pin  
0110: PG[11] pin  
0111: PH[11] pin  
1000: PI[11] pin

Bits 11:8 **EXTI10[3:0]**: EXTI 10 configuration bits

These bits are written by software to select the source input for the EXTI10 external interrupt.

0000: PA[10] pin  
0001: PB[10] pin  
0010: PC[10] pin  
0011: PD[10] pin  
0100: PE[10] pin  
0101: PF[10] pin  
0110: PG[10] pin  
0111: PH[10] pin  
1000: PI[10] pin

Bits 7:4 **EXTI9[3:0]**: EXTI 9 configuration bits

These bits are written by software to select the source input for the EXTI9 external interrupt.

0000: PA[9] pin  
0001: PB[9] pin  
0010: PC[9] pin  
0011: PD[9] pin  
0100: PE[9] pin  
0101: PF[9] pin  
0110: PG[9] pin  
0111: PH[9] pin  
1000: PI[9] pin

Bits 3:0 **EXTI8[3:0]**: EXTI 8 configuration bits

These bits are written by software to select the source input for the EXTI8 external interrupt.

0000: PA[8] pin  
0001: PB[8] pin  
0010: PC[8] pin  
0011: PD[8] pin  
0100: PE[8] pin  
0101: PF[8] pin  
0110: PG[8] pin  
0111: PH[8] pin  
1000: PI[8] pin

*Note:* Some of the I/O pins mentioned in the above register may not be available on small packages.

**9.2.6 SYSCFG external interrupt configuration register 4 (SYSCFG\_EXTICR4)**

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 15:12 **EXTI15[3:0]**: EXTI 15 configuration bits

These bits are written by software to select the source input for the EXTI15 external interrupt.

- 0000: PA[15] pin
- 0001: PB[15] pin
- 0010: PC[15] pin
- 0011: PD[15] pin
- 0100: PE[15] pin
- 0101: PF[15] pin
- 0110: PG[15] pin
- 0111: PH[15] pin
- 1000: Reserved

Bits 11:8 **EXTI14[3:0]**: EXTI 14 configuration bits

These bits are written by software to select the source input for the EXTI14 external interrupt.

- 0000: PA[14] pin
- 0001: PB[14] pin
- 0010: PC[14] pin
- 0011: PD[14] pin
- 0100: PE[14] pin
- 0101: PF[14] pin
- 0110: PG[14] pin
- 0111: PH[14] pin
- 1000: Reserved

Bits 7:4 **EXTI13[3:0]**: EXTI 13 configuration bits

These bits are written by software to select the source input for the EXTI13 external interrupt.

- 0000: PA[13] pin
- 0001: PB[13] pin
- 0010: PC[13] pin
- 0011: PD[13] pin
- 0100: PE[13] pin
- 0101: PF[13] pin
- 0110: PG[13] pin
- 0111: PH[13] pin
- 1000: Reserved

Bits 3:0 **EXTI12[3:0]**: EXTI 12 configuration bits

These bits are written by software to select the source input for the EXTI12 external interrupt.

- 0000: PA[12] pin
- 0001: PB[12] pin
- 0010: PC[12] pin
- 0011: PD[12] pin
- 0100: PE[12] pin
- 0101: PF[12] pin
- 0110: PG[12] pin
- 0111: PH[12] pin
- 1000: Reserved

*Note:* Some of the I/O pins mentioned in the above register may not be available on small packages.

### 9.2.7 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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	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	SRAM2 BSY	SRAM2 ER
														r	rw



Bits 31:2 Reserved, must be kept at reset value

Bit 1 **SRAM2BSY**: SRAM2 busy by erase operation

- 0: No SRAM2 erase operation is on going.
- 1: SRAM2 erase operation is on going.

Bit 0 **SRAM2ER**: SRAM2 Erase

Setting this bit starts a hardware SRAM2 erase operation. This bit is automatically cleared at the end of the SRAM2 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.*

### 9.2.8 SYSCFG configuration register 2 (SYSCFG\_CFGR2)

Address offset: 0x1C

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	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 TIM1/8/15/16/17 Break input.

- 0: ECC error disconnected from TIM1/8/15/16/17 Break input.
- 1: ECC error connected to TIM1/8/15/16/17 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 TIM1/8/15/16/17 Break input, as well as the PVDE and PLS[2:0] in the PWR\_CR2 register.

0: PVD interrupt disconnected from TIM1/8/15/16/17 Break input. PVDE and PLS[2:0] bits can be programmed by the application.

1: PVD interrupt connected to TIM1/8/15/16/17 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 TIM1/8/15/16/17 Break inputs.

0: SRAM2 parity error signal disconnected from TIM1/8/15/16/17 Break inputs

1: SRAM2 parity error signal connected to TIM1/8/15/16/17 Break inputs

Bit 0 **CLL**: Cortex<sup>®</sup>-M4 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 Cortex<sup>®</sup>-M4 LOCKUP (Hardfault) output to TIM1/8/15/16/17 Break input

0: Cortex<sup>®</sup>-M4 LOCKUP output disconnected from TIM1/8/15/16/17 Break inputs

1: Cortex<sup>®</sup>-M4 LOCKUP output connected to TIM1/8/15/16/17 Break inputs

### 9.2.9 SYSCFG SRAM2 write protection register (SYSCFG\_SWPR)

Address offset: 0x20

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	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	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	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **PxWP** (x = 0 to 31): SRAM2 page x write protection

These bits are set by software and cleared only by a system reset.

0: Write protection of SRAM2 page x is disabled.

1: Write protection of SRAM2 page x is enabled.

### 9.2.10 SYSCFG SRAM2 key register (SYSCFG\_SKR)

Address offset: 0x24

System reset value: 0x0000 0000

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

### 9.2.11 SYSCFG SRAM2 write protection register 2 (SYSCFG\_SWPR2)

Address offset: 0x28

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	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	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	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **PxWP** (x= 32 to 63): SRAM2 page x write protection

These bits are set by software and cleared only by a system reset.

- 0: Write protection of SRAM2 page x is disabled.
- 1: Write protection of SRAM2 page x is enabled.

9.2.12 SYSCFG register map

The following table gives the SYSCFG register map and the reset values.

Table 42. SYSCFG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	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.	MEM_MODE		
	Reset value																								0	0					x	x	x	
0x04	SYSCFG_CFGR1	FPU_IE[5..0]							Res.	Res.	I2C4_FMP	I2C3_FMP	I2C2_FMP	I2C1_FMP	I2C_PB9_FMP	I2C_PB8_FMP	I2C_PB7_FMP	I2C_PB6_FMP	Res.	Res.	Res.	Res.	Res.	ANAS/WDD	BOOSTEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FWDIS	
	Reset value	0	1	1	1	1	1				0	0	0	0	0	0	0	0						0	0					0	0	0	1	
0x08	SYSCFG_EXTICR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI3 [3:0]	EXTI2 [3:0]	EXTI1 [3:0]	EXTI0 [3:0]	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		
0x0C	SYSCFG_EXTICR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI7 [3:0]	EXTI6 [3:0]	EXTI5 [3:0]	EXTI4 [3:0]	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		
0x10	SYSCFG_EXTICR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI11 [3:0]	EXTI10 [3:0]	EXTI9 [3:0]	EXTI8 [3:0]	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		
0x14	SYSCFG_EXTICR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI15 [3:0]	EXTI14 [3:0]	EXTI13 [3:0]	EXTI12 [3:0]	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		
0x18	SYSCFG_SCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM2BS	SRAM2ER	
	Reset value																														0	0		
0x1C	SYSCFG_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCL	PVDL
	Reset value																								0							0	0	0
0x20	SYSCFG_SWPR	P31WP	P30WP	P29WP	P28WP	P27WP	P26WP	P25WP	P24WP	P23WP	P22WP	P21WP	P20WP	P19WP	P18WP	P17WP	P16WP	P15WP	P14WP	P13WP	P12WP	P11WP	P10WP	P9WP	P8WP	P7WP	P6WP	P5WP	P4WP	P3WP	P2WP	P1WP	P0WP	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x24	SYSCFG_SKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY
	Reset value																																	
0x29	SYSCFG_SWPR2	P63WP	P62WP	P61WP	P60WP	P59WP	P58WP	P57WP	P56WP	P55WP	P54WP	P53WP	P52WP	P51WP	P50WP	P49WP	P48WP	P47WP	P46WP	P45WP	P44WP	P43WP	P42WP	P41WP	P40WP	P39WP	P38WP	P37WP	P36WP	P35WP	P34WP	P33WP	P32WP	
	Reset value	0	0	0	0	0	0	0	0	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 2.2.2 on page 82](#) for the register boundary addresses.

# 10 Peripherals interconnect matrix

## 10.1 Introduction

Several peripherals have direct connections between them.

This allows autonomous communication and or synchronization between peripherals, saving CPU resources thus power supply consumption.

In addition, these hardware connections remove software latency and allow design of predictable system.

Depending on peripherals, these interconnections can operate in Run, Sleep, Low-power run and sleep, Stop 0, Stop 1 and Stop 2 modes.

## 10.2 Connection summary

Table 43. STM32L4Rxxx and STM32L4Sxxx peripherals interconnect matrix<sup>(1) (2)</sup>

Source	Destination																					
	TIM1	TIM8	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM15	TIM16	TIM17	LPTIM1	LPTIM2	ADC1	DFSDM1	OPAMP1	OPAMP2	DAC1	DAC2	COMP1	COMP2	IRTIM
TIM1	-	1	1	1	1	-	-	-	1	-	-	-	-	2	5	-	-	-	-	9	-	-
TIM8	-	-	1	-	1	1	-	-	-	-	-	-	-	2	5	-	-	4	4	-	9	-
TIM2	1	1	-	1	1	1	-	-	-	-	-	-	-	2	-	-	-	4	4	9	-	-
TIM3	1	-	1	-	1	1	-	-	1	-	-	-	-	2	5	-	-	-	-	9	9	-
TIM4	1	1	1	1	-	1	-	-	-	-	-	-	-	2	5	-	-	4	4	-	-	-
TIM5	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	4	-	-	-
TIM6	-	-	-	-	-	-	-	-	-	-	-	-	-	2	5	-	-	4	4	-	-	-
TIM7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	5	-	-	4	4	-	-	-
TIM15	1	-	-	1	-	-	-	-	-	-	-	-	-	2	-	-	-	-	-	-	9	-
TIM16	-	-	-	-	-	-	-	-	1	-	-	-	-	-	5	-	-	-	-	-	-	15
TIM17	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	15
LPTIM1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	5	-	-	-	-	-	-	-
LPTIM2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	5	-	-	-	-	-	-	-
ADC1	3	-	-	-	-	-	-	-	-	-	-	-	-	-	16	-	-	-	-	-	-	-
DFSDM1	6	6	-	-	-	-	-	-	6	6	6	6	6	-	-	-	-	-	-	-	-	-
T. Sensor	-	-	-	-	-	-	-	-	-	-	-	-	-	12	-	-	-	-	-	-	-	-
VBAT	-	-	-	-	-	-	-	-	-	-	-	-	-	12	-	-	-	-	-	-	-	-
VREFINT	-	-	-	-	-	-	-	-	-	-	-	-	-	12	-	-	-	-	-	-	-	-
OPAMP1	-	-	-	-	-	-	-	-	-	-	-	-	-	12	-	-	-	-	-	-	-	-

Table 43. STM32L4Rxxx and STM32L4Sxxx peripherals interconnect matrix<sup>(1) (2)</sup>

Source	Destination																						
	TIM1	TIM8	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM15	TIM16	TIM17	LPTIM1	LPTIM2	ADC1	DFSDM1	OPAMP1	OPAMP2	DAC1	DAC2	COMP1	COMP2	IRTIM	
OPAMP2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	12	-	-	-	-	-	-	-	-
DAC1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	12	12	-	-	-	-	-	-
DAC2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
HSE	-	-	-	-	-	-	-	-	-	-	7	-	-	-	-	-	-	-	-	-	-	-	-
LSE	-	-	7	-	-	-	-	-	7	7	-	-	-	-	-	-	-	-	-	-	-	-	-
MSI	-	-	-	-	-	-	-	-	-	-	7	-	-	-	-	-	-	-	-	-	-	-	-
LSI	-	-	-	-	-	-	-	-	7	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MCO	-	-	-	-	-	-	-	-	-	-	7	-	-	-	-	-	-	-	-	-	-	-	-
EXTI	-	-	-	-	-	-	-	-	-	-	-	-	-	2	5	-	-	4	4	-	-	-	-
RTC	-	-	-	-	-	-	-	-	7	-	8	8	-	-	-	-	-	-	-	-	-	-	-
COMP1	13	13	13	13	-	-	-	-	13	13	13	8	8	-	-	-	-	-	-	-	-	-	-
COMP2	13	13	13	13	-	-	-	-	13	13	13	8	8	-	-	-	-	-	-	-	-	-	-
SYST ERR	14	14	-	-	-	-	-	-	14	14	14	-	-	-	-	-	-	-	-	-	-	-	-
USB	-	-	11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

1. Numbers in table are links to corresponding detailed sub-section in [Section 10.3: Interconnection details](#).
2. The “-” symbol in grayed cells means no interconnect.

### 10.3 Interconnection details

#### 10.3.1 From timer (TIM1/TIM2/TIM3/TIM4/TIM5/TIM8/TIM15/TIM16/TIM17) to timer (TIM1/TIM2/TIM3/TIM4/TIM5/TIM8/TIM15)

##### Purpose

Some of the TIMx timers are linked together internally for timer synchronization or chaining.

When one timer is configured in Master Mode, it can reset, start, stop or clock the counter of another timer configured in Slave Mode.

A description of the feature is provided in: [Section 35.3.19: Timer synchronization](#).

The modes of synchronization are detailed in:

- [Section 34.3.26: Timer synchronization](#) for advanced-control timers (TIM1/TIM8)
- [Section 35.3.18: Timers and external trigger synchronization](#) for general-purpose timers (TIM2/TIM3/TIM4/TIM5)
- [Section 36.4.19: External trigger synchronization \(TIM15 only\)](#) for general-purpose timer (TIM15)

### Triggering signals

The output (from Master) is on signal TIMx\_TRGO (and TIMx\_TRGO2 for TIM1/TIM8) following a configurable timer event.

The input (to slave) is on signals TIMx\_ITR0/ITR1/ITR2/ITR3

The input and output signals for TIM1/TIM8 are shown in [Figure 255: Advanced-control timer block diagram](#).

The possible master/slave connections are given in:

- [Table 217: TIMx internal trigger connection](#)
- [Table 222: TIMx internal trigger connection](#)
- [Table 226: TIMx Internal trigger connection](#)

### Active power mode

Run, Sleep, Low-power run, Low-power sleep.

## 10.3.2 From timer (TIM1/TIM2/TIM3/TIM4/TIM6/TIM8/TIM15) and EXTI to ADC (ADC1)

### Purpose

General-purpose timers (TIM2/TIM3/TIM4), basic timer (TIM6), advanced-control timers (TIM1/TIM8), general-purpose timer (TIM15) and EXTI can be used to generate an ADC triggering event.

TIMx synchronization is described in: [Section 34.3.27: ADC synchronization](#) (TIM1/TIM8).

ADC synchronization is described in: [Section 21.4.18: Conversion on external trigger and trigger polarity \(EXTSEL, EXTEN, JEXTSEL, JEXTEN\)](#).

### Triggering signals

The output (from timer) is on signal TIMx\_TRGO, TIMx\_TRGO2 or TIMx\_CCx event.

The input (to ADC) is on signal EXT[15:0], JEXT[15:0].

The connection between timers and ADC is provided in:

- [Table 118: ADC1 - External triggers for regular channels](#)
- [Table 119: ADC1 - External trigger for injected channels](#)

### Active power mode

Run, Sleep, Low-power run, Low-power sleep.

## 10.3.3 From ADC to timer (TIM1/TIM8)

### Purpose

ADC1 can provide trigger event through watchdog signals to advanced-control timers (TIM1/TIM8).

A description of the ADC analog watchdog setting is provided in: [Section 21.4.29: Analog window watchdog \(AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD\\_HTx, AWD\\_LTx, AWDx\)](#).

Trigger settings on the timer are provided in: [Section 34.3.4: External trigger input](#).

### Triggering signals

The output (from ADC) is on signals ADC\_AWDx\_OUT n = 1 (for ADC1) x = 1, 2, 3 (3 watchdog per ADC) and the input (to timer) on signal TIMx\_ETR (external trigger).

### Active power mode

Run, Sleep, Low-power run, Low-power sleep.

## 10.3.4 From timer (TIM2/TIM4/TIM5/TIM6/TIM7/TIM8) and EXTI to DAC (DAC1/DAC2)

### Purpose

General-purpose timers (TIM2/TIM4/TIM5), basic timers (TIM6, TIM7), advanced-control timers (TIM8) and EXTI can be used as triggering event to start a DAC conversion.

### Triggering signals

The output (from timer) is on signal TIMx\_TRGO directly connected to corresponding DAC inputs.

Selection of input triggers on DAC is provided in [Section 22.4.6: DAC trigger selection](#) (single and dual mode).

### Active power mode

Run, Sleep, Low-power run, Low-power sleep.

## 10.3.5 From timer (TIM1/TIM3/TIM4/TIM6/TIM7/TIM8/TIM16/LPTIM1/LPTIM2) and EXTI to DFSDM1

### Purpose

General-purpose timers (TIM3/TIM4), basic timers (TIM6/TIM7), advanced-control timers (TIM1/TIM8), general-purpose timer (TIM16), low power timers (LPTIM1/LPTIM2) and EXTI can be used to generate a triggering event on DFSDM1 module (on each possible data block DFSDM1\_FLT0/DFSDM1\_FLT1/DFSDM1\_FLT2/DFSDM1\_FLT3) and start an ADC conversion.

DFSDM triggered conversion feature is described in: [Section 27.4.15: Launching conversions](#).

### Triggering signals

The output (from timer) is on signal TIMx\_TRGO/TIMx\_TRGO2 or TIM16\_OC1.

The input (on DFSDM1) is on signal DFSDM1\_INTRG[0:8].

The connection between timers, EXTI and DFSDM1 is provided in [Table 164: DFSDM triggers connection](#).

### Active power mode

Run, Sleep, Low-power run, Low-power sleep.

### 10.3.6 From DFSDM1 to timer (TIM1/TIM8/TIM15/TIM16/TIM17)

#### Purpose

DFSDM1 can generate a timer break on advanced-control timers (TIM1/TIM8) and general-purpose timers (TIM15/TIM16/TIM17) when a watchdog is activated (minimum or maximum threshold value crossed by analog signal) or when a short-circuit detection is made.

DFSDM1 watchdog is described in [Section 27.4.10: Analog watchdog](#).

DFSDM1 short-circuit detection is described in [Section 27.4.11: Short-circuit detector](#).

Timer break is described in:

- [Section 34.3.16: Using the break function](#) (TIM1/TIM8)
- [Section 36.4.13: Using the break function](#) (TIM15/TIM16/TIM17)

#### Triggering signals

The output (from DFSDM1) is on signals `dfsdm1_break[0:3]` directly connected to timer and 'Ored' with other break input signals of the timer.

#### Active power mode

Run, Sleep, Low-power run, Low-power sleep.

### 10.3.7 From HSE, LSE, LSI, MSI, MCO, RTC to timer (TIM2/TIM15/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 timer (TIM15/16/17) channel 1.

This allows to calibrate the HSI16/MSI system clocks (with TIM15/TIM16 and LSE) or 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 6.2.18: Internal/external clock measurement with TIM15/TIM16/TIM17](#).

External clock LSE can be used as input to general-purpose timers (TIM2) on TIM2\_ETR pin, see [Section 35.4.19: TIM2 option register 1 \(TIM2\\_OR1\)](#).

#### Active power mode

Run, Sleep, Low-power run, Low-power sleep.

### 10.3.8 From RTC, COMP1, COMP2 to low-power timer (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 38.4.6: Trigger multiplexer](#) (and following sections).

The input selection is described in [Table 233: LPTIM1 external trigger connection](#).

### Active power mode

Run, Sleep, Low-power run, Low-power sleep, Stop 0, Stop 1, Stop 2 (LPTIM1 only).

## 10.3.9 From timer (TIM1/TIM2/TIM3/TIM8/TIM15) to comparators (COMP1/COMP2)

### Purpose

Advanced-control timers (TIM1/TIM8), general-purpose timers (TIM2/TIM3) and general-purpose timer (TIM15) can be used as blanking window input to COMP1/COMP2

The blanking function is described in [Section 25.3.7: Comparator output blanking function](#).

The blanking sources are given in:

- [Section 25.6.1: Comparator 1 control and status register \(COMP1\\_CSR\)](#) bits 20:18 BLANKING[2:0]
- [Section 25.6.2: 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

Run, Sleep, Low-power run, Low-power sleep.

## 10.3.10 From USB to timer (TIM2)

### Purpose

USB (OTG\_FS SOF) can generate a trigger to general-purpose timer (TIM2).

Connection of USB to TIM2 is described in [Table 222: TIMx internal trigger connection](#).

### Triggering signals

Internal signal generated by USB FS Start Of Frame.

### Active power mode

Run, Sleep.

### 10.3.11 From internal analog source to ADC and OPAMP (OPAMP1/OPAMP2)

#### Purpose

Internal temperature sensor ( $V_{TS}$ ) and  $V_{BAT}$  monitoring channel are connected to ADC1 input channels.

Internal reference voltage ( $V_{REFINT}$ ) is connected to ADC1 input channels.

OPAMP1 and OPAMP2 outputs can be connected to ADC1 input channels through the GPIO.

DAC1\_OUT1 can be connected to OPAMP1\_VINP.

DAC1\_OUT2 can be connected to OPAMP2\_VINP.

This is according:

- [Section 21.2: ADC main features](#)
- [Section 21.4.11: Channel selection \(SQRx, JSQRx\)](#)
- [Section Figure 86.: ADC1 connectivity](#)
- [Table 85: Operational amplifier possible connections](#)

#### Active power mode

Run, Sleep, Low-power run, Low-power sleep.

### 10.3.12 From comparators (COMP1/COMP2) to timers (TIM1/TIM2/TIM3/TIM8/TIM15/TIM16/TIM17)

#### Purpose

Comparators (COMP1/COMP2) output values can be connected to timers (TIM1/TIM2/TIM3/TIM8/TIM15/TIM16/TIM17) input captures or TIMx\_ETR signals.

The connection to ETR is described in [Section 34.3.4: External trigger input](#).

Comparators (COMP1/COMP2) output values can also generate break input signals for timers (TIM1/TIM8) on input pins TIMx\_BKIN or TIMx\_BKIN2 through GPIO alternate function selection using open drain connection of IO, see [Section 34.3.17: Bidirectional break inputs](#).

The possible connections are given in:

- [Section 34.4.21: TIM1 option register 1 \(TIM1\\_OR1\)](#)
- [Section 34.4.22: TIM8 option register 1 \(TIM8\\_OR1\)](#)
- [Section 35.4.19: TIM2 option register 1 \(TIM2\\_OR1\)](#)
- [Section 35.4.20: TIM3 option register 1 \(TIM3\\_OR1\)](#)
- [Section 35.4.23: TIMx register map](#)
- [Section 35.4.22: TIM3 option register 2 \(TIM3\\_OR2\)](#)
- [Section 36.3: TIM16/TIM17 main features](#)

#### Active power mode

Run, Sleep, Low-power run, Low-power sleep.



### 10.3.13 From system errors to timers (TIM1/TIM8/TIM15/TIM16/TIM17)

#### Purpose

CSS, CPU hardfault, RAM parity error, FLASH ECC double error detection, PVD can generate system errors in the form of timer break toward timers (TIM1/TIM8/TIM15/TIM16/TIM17).

The purpose of the break function is to protect power switches driven by PWM signals generated by the timers.

List of possible source of break are described in:

- [Section 34.3.16: Using the break function](#) (TIM1/TIM8)
- [Section 36.4.13: Using the break function](#) (TIM15/TIM16/TIM17)
- [Figure 365: TIM15 block diagram](#)
- [Figure 366: TIM16/TIM17 block diagram](#)

#### Active power mode

Run, Sleep, Low-power run, Low-power sleep.

### 10.3.14 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 39: Infrared interface \(IRTIM\)](#).

#### Active power mode

Run, Sleep, Low-power run, Low-power sleep.

### 10.3.15 From ADC (ADC1) to DFSDM

#### Purpose

Up to 3 internal ADC results can be directly connected through a parallel bus to DFSDM input in order to use DFSDM filtering capabilities.

The feature is described as part of DFSDM peripheral description in [Section 27.4.6: Parallel data inputs - Input from internal ADC](#)

The possible connections are given in:

- [Section 27.7.1: DFSDM channel y configuration register \(DFSDM\\_CHyCFGR1\)](#)
  - Bits 13:12 DATMPX[1:0]: Input data multiplexer for channel y
- [Section 27.7.5: DFSDM channel y data input register \(DFSDM\\_CHyDATINR\)](#)
  - Bits 31:16 INDAT0[15:0]: Input data for channel y or channel y+1
  - Bits 15:0 INDAT0[15:0]: Input data for channel y

#### Active power mode

Run, Sleep, Low-power run, Low-power sleep.

# 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 (7 channels) and DMA2 (7 channels).

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 [Table 44](#).

**Table 44. 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.

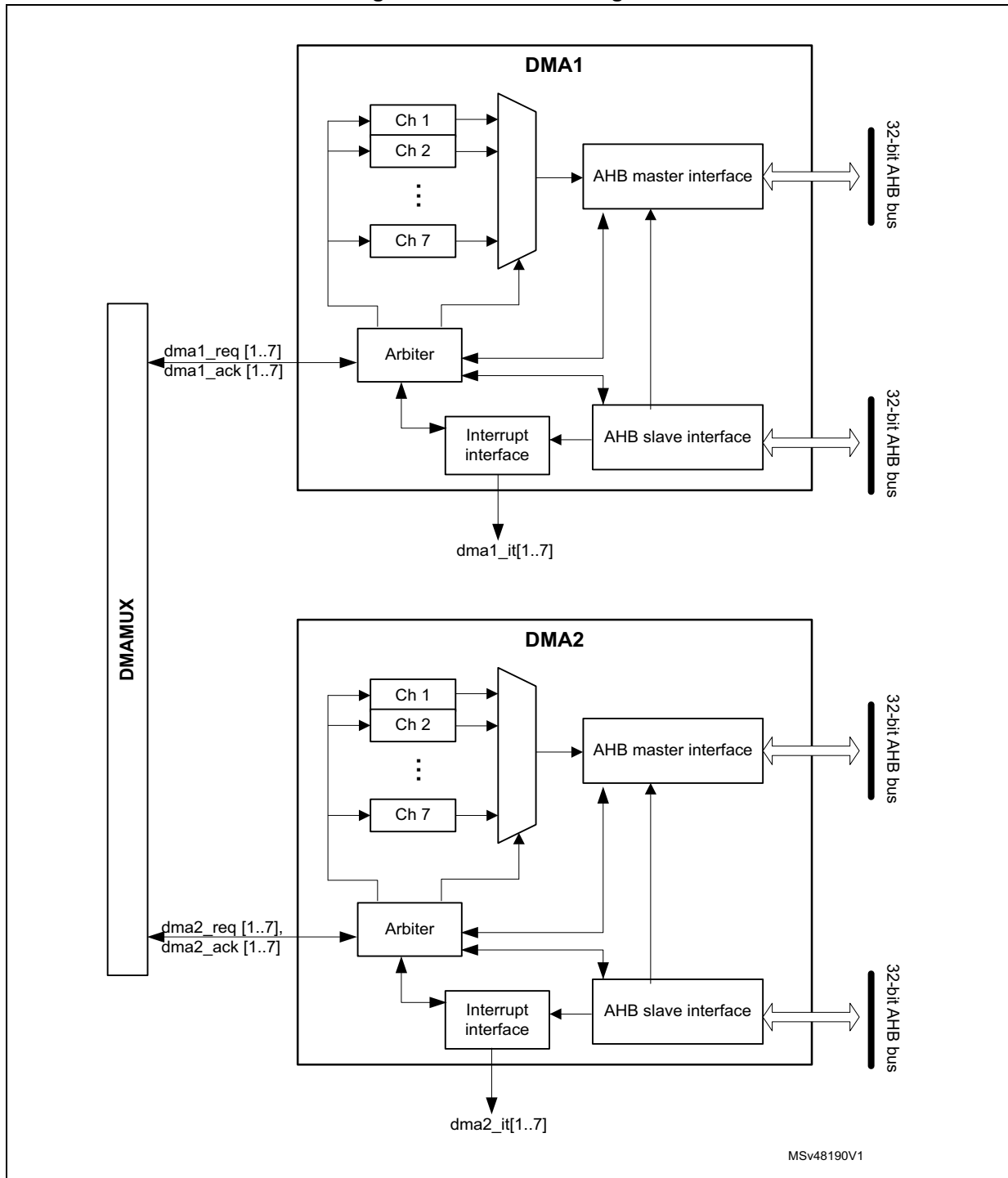
For the mapping of the different requests, refer to the [Section 12.3: DMAMUX implementation](#).

## 11.4 DMA functional description

### 11.4.1 DMA block diagram

The DMA block diagram is shown in [Figure 28](#).

Figure 28. 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 45. 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.*

*When a channel is enabled and still active (not completed), the software must perform two separate write accesses to the DMA\_CCRx register, to disable the channel, then to*

*reprogram the channel for another next block transfer.  
Some fields of the DMA\_CCRx register are read-only when the EN bit is set to 1.*

### Stop and resume a channel

Once the software activates a channel, it waits for the completion of the programmed transfer. The DMA controller is not able to resume an aborted active channel with a possible suspended bus transfer.

To correctly stop and disable a channel, the software clears the EN bit of the DMA\_CCRx register. The software secures that no pending request from the peripheral is served by the DMA controller before the transfer completion. The software waits for the transfer complete or transfer error interrupt.

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 [Table 46](#).

**Table 46. 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	8	@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 47. 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\_CNDTRx)

Address offset:  $0x0C + 0x14 * (x - 1)$ , ( $x = 1$  to  $7$ )

Reset value:  $0x0000\ 0000$

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

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

Bits 15:0 **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\_CCRx register).
- It is reloaded automatically by the previously programmed value, when the transfer is complete, if the channel is in circular mode (CIRC = 1).

If this 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\_CPARx)

Address offset:  $0x10 + 0x14 * (x - 1)$ , ( $x = 1$  to  $7$ )

Reset value:  $0x0000\ 0000$

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

Bits 31:0 **PA[31:0]**: peripheral address

It contains the base address of the peripheral data register from/to which the data will be 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 will be 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 and reset values

Table 48 gives the DMA register map and reset values.

Table 48. 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



Table 48. DMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	DMA_IFCR	Res.	Res.	Res.	Res.	CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5	CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DMA_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]	MSIZE[1:0]	PSIZE[1:0]	MSIZE[1:0]	PSIZE[1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN	
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	
0x00C	DMA_CNDTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x010	DMA_CPAR1	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DMA_CMAR1	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	Reserved	Reserved																															
0x01C	DMA_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]	MSIZE[1:0]	PSIZE[1:0]	MSIZE[1:0]	PSIZE[1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN	
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	
0x020	DMA_CNDTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x024	DMA_CPAR2	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	DMA_CMAR2	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	Reserved	Reserved																															
0x030	DMA_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]	MSIZE[1:0]	PSIZE[1:0]	MSIZE[1:0]	PSIZE[1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN	
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	
0x034	DMA_CNDTR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x038	DMA_CPAR3	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	DMA_CMAR3	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	Reserved	Reserved																															
0x044	DMA_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]	MSIZE[1:0]	PSIZE[1:0]	MSIZE[1:0]	PSIZE[1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN	
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	
0x048	DMA_CNDTR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x04C	DMA_CPAR4	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	DMA_CMAR4	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	Reserved	Reserved																															

Table 48. DMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x058	DMA_CCR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]	MSIZE[1:0]	PSIZE[1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x05C	DMA_CNDTR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x060	DMA_CPAR5	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	DMA_CMAR5	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	Reserved	Reserved.																															
0x06C	DMA_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]	MSIZE[1:0]	PSIZE[1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x070	DMA_CNDTR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x074	DMA_CPAR6	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x078	DMA_CMAR6	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x07C	Reserved	Reserved.																															
0x080	DMA_CCR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]	MSIZE[1:0]	PSIZE[1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x084	DMA_CNDTR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x088	DMA_CPAR7	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x08C	DMA_CMAR7	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Refer to [Section 2.2.2](#) for 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 de-asserted.

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
- 26 trigger inputs to DMA request generator
- 26 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:
  - 89 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

### 12.3.1 DMAMUX instantiation

DMAMUX is instantiated with the hardware configuration parameters listed in the following table.

**Table 49. DMAMUX instantiation**

Feature	DMAMUX
Number of DMAMUX output request channels	14
Number of DMAMUX request generator channels	4
Number of DMAMUX request trigger inputs	26
Number of DMAMUX synchronization inputs	26
Number of DMAMUX peripheral request inputs	89

### 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 0 to 6
- DMAMUX channels 7 to 13 are connected to DMA2 channels 0 to 6

**Table 50. 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	44	TIM1_CH3	87	DFSDM1_FLT1
2	dmamux_req_gen1	45	TIM1_CH4	88	DFSDM1_FLT2
3	dmamux_req_gen2	46	TIM1_UP	89	DFSDM1_FLT3
4	dmamux_req_gen3	47	TIM1_TRIG	90	DCMI
5	ADC1	48	TIM1_COM	91	AES_IN
6	DAC1	49	TIM8_CH1	92	AES_OUT
7	DAC2	50	TIM8_CH2	93	HASH_IN
8	TIM6_UP	51	TIM8_CH3	94	Reserved
9	TIM7_UP	52	TIM8_CH4	95	Reserved
10	SPI1_RX	53	TIM8_UP	96	Reserved
11	SPI1_TX	54	TIM8_TRIG	97	Reserved
12	SPI2_RX	55	TIM8_COM	98	Reserved
13	SPI2_TX	56	TIM2_CH1	99	Reserved
14	SPI3_RX	57	TIM2_CH2	100	Reserved
15	SPI3_TX	58	TIM2_CH3	101	Reserved
16	I2C1_RX	59	TIM2_CH4	102	Reserved



Table 50. DMAMUX: assignment of multiplexer inputs to resources (continued)

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
17	I2C1_TX	60	TIM2_UP	103	Reserved
18	I2C2_RX	61	TIM3_CH1	104	Reserved
19	I2C2_TX	62	TIM3_CH2	105	Reserved
20	I2C3_RX	63	TIM3_CH3	106	Reserved
21	I2C3_TX	64	TIM3_CH4	107	Reserved
22	I2C4_RX	65	TIM3_UP	108	Reserved
23	I2C4_TX	66	TIM3_TRIG	109	Reserved
24	USART1_RX	67	TIM4_CH1	110	Reserved
25	USART1_TX	68	TIM4_CH2	111	Reserved
26	USART2_RX	69	TIM4_CH3	112	Reserved
27	USART2_TX	70	TIM4_CH4	113	Reserved
28	USART3_RX	71	TIM4_UP	114	Reserved
29	USART3_TX	72	TIM5_CH1	115	Reserved
30	UART4_RX	73	TIM5_CH2	116	Reserved
31	UART4_TX	74	TIM5_CH3	117	Reserved
32	UART5_RX	75	TIM5_CH4	118	Reserved
33	UART5_TX	76	TIM5_UP	119	Reserved
34	LPUART1_RX	77	TIM5_TRIG	120	Reserved
35	LPUART1_TX	78	TIM15_CH1	121	Reserved
36	SAI1_A	79	TIM15_UP	122	Reserved
37	SAI1_B	80	TIM15_TRIG	123	Reserved
38	SAI2_A	81	TIM15_COM	124	Reserved
39	SAI2_B	82	TIM16_CH1	125	Reserved
40	OCTOSPI1	83	TIM16_UP	126	Reserved
41	OCTOSPI2	84	TIM17_CH1	127	Reserved
42	TIM1_CH1	85	TIM17_UP	-	-
43	TIM1_CH2	86	DFSDM1_FLT0	-	-

Table 51. 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	dmamux_evt2
3	EXTI LINE3	19	dmamux_evt3
4	EXTI LINE4	20	LPTIM1_OUT
5	EXTI LINE5	21	LPTIM2_OUT

Table 51. DMAMUX: assignment of trigger inputs to resources (continued)

Trigger input	Resource	Trigger input	Resource
6	EXTI LINE6	22	DSI Tearing Effect
7	EXTI LINE7	23	DSI End of refresh
8	EXTI LINE8	24	DMA2D End of Transfer
9	EXTI LINE9	25	LTDC Line interrupt
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 52. DMAMUX: assignment of synchronization inputs to resources

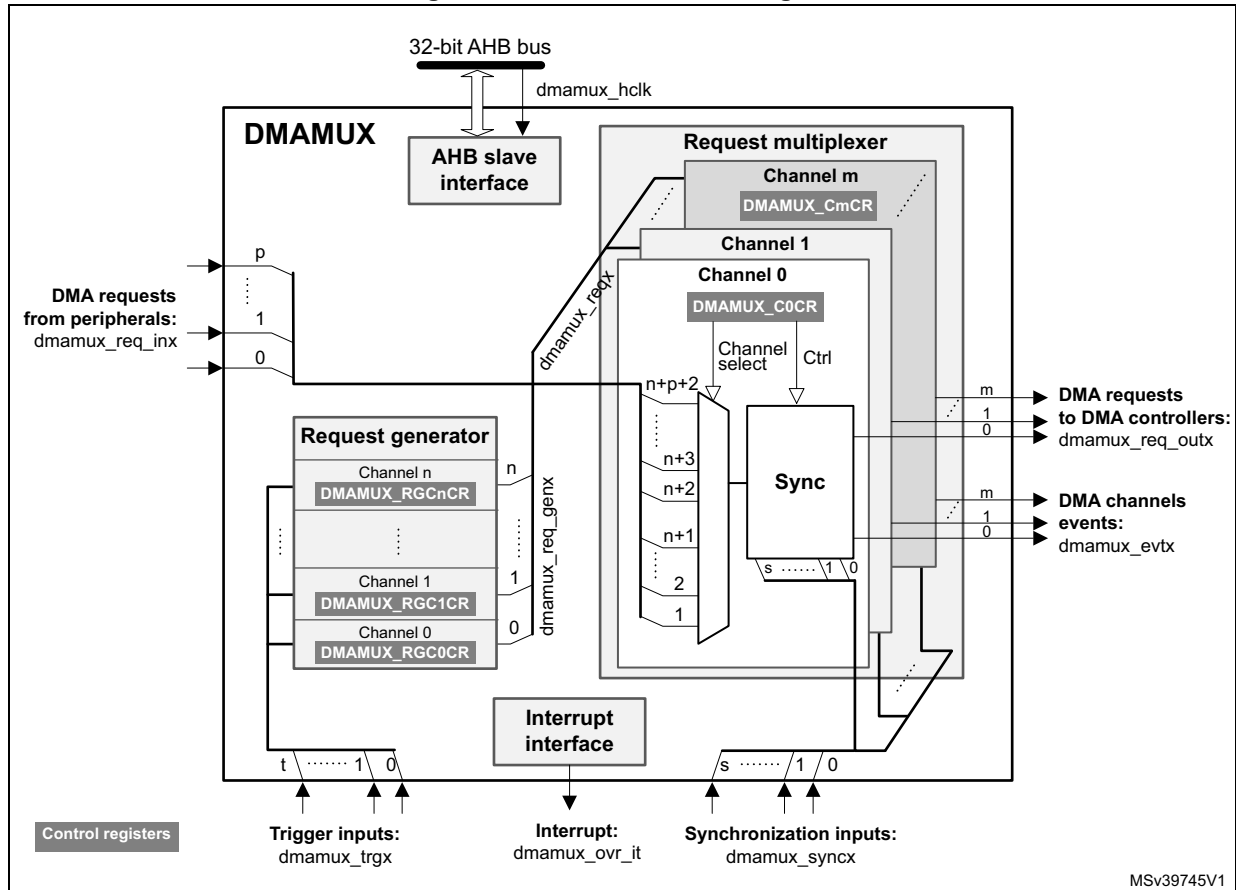
Sync. input	Resource	Sync. input	Resource
0	EXTI LINE0	16	dmamux_evt0
1	EXTI LINE1	17	dmamux_evt1
2	EXTI LINE2	18	dmamux_evt2
3	EXTI LINE3	19	dmamux_evt3
4	EXTI LINE4	20	LPTIM1_OUT
5	EXTI LINE5	21	LPTIM2_OUT
6	EXTI LINE6	22	DSI Tearing Effect
7	EXTI LINE7	23	DSI End of refresh
8	EXTI LINE8	24	DMA2D End of Transfer
9	EXTI LINE9	25	LTDC Line interrupt
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 29 shows the DMAMUX block diagram.

Figure 29. DMAMUX block diagram



DMAMUX features two main sub-blocks: the request line multiplexer and the request line generator.

The implementation assigns:

- DMAMUX request multiplexer sub-block inputs (`dmamux_reqx`) from peripherals (`dmamux_req_inx`) and from channels of the DMAMUX request generator sub-block (`dmamux_req_genx`)
- DMAMUX request outputs to channels of DMA controllers (`dmamux_req_outx`)
- Internal or external signals to DMA request trigger inputs (`dmamux_trgx`)
- Internal or external signals to synchronization inputs (`dmamux_syncx`)

## 12.4.2 DMAMUX signals

Table 53 lists the DMAMUX signals.

**Table 53. 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

The following sequence should be followed to configure both a DMAMUX x channel and the related DMA channel y:

## 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. It is not allowed to configure a same non-null DMAREQ\_ID to two different channels of the DMAMUX request line multiplexer.

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 31](#), 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 will be discarded. The following asserted input request lines will not be 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 de-asserted), 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 30. Synchronization mode of the DMAMUX request line multiplexer channel

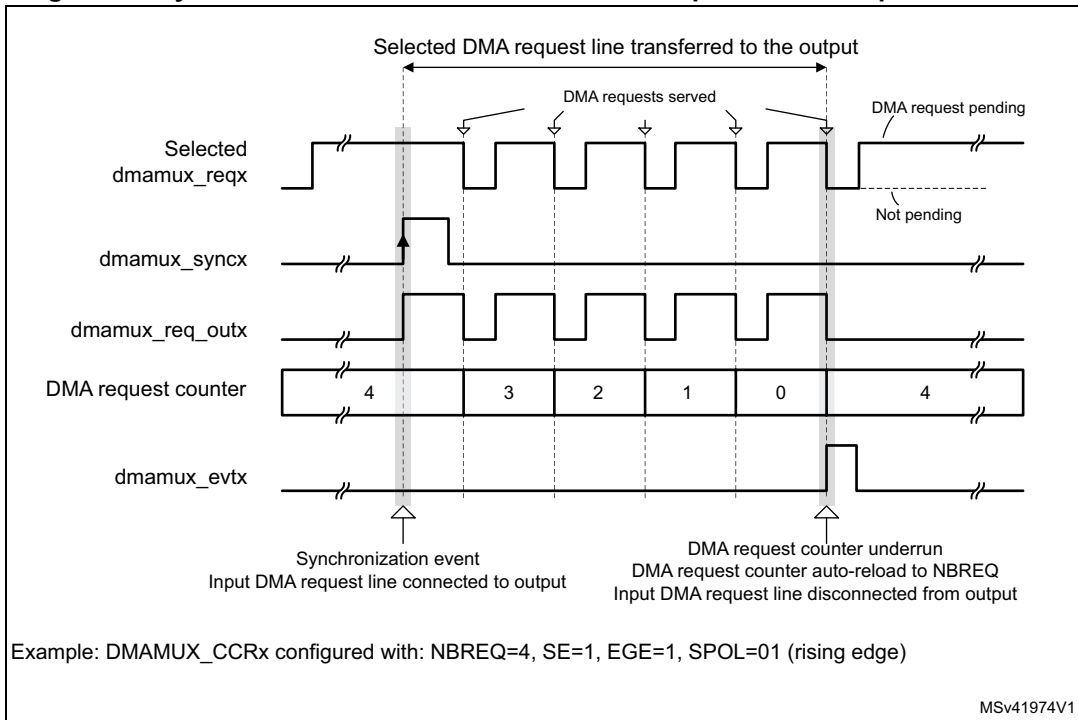
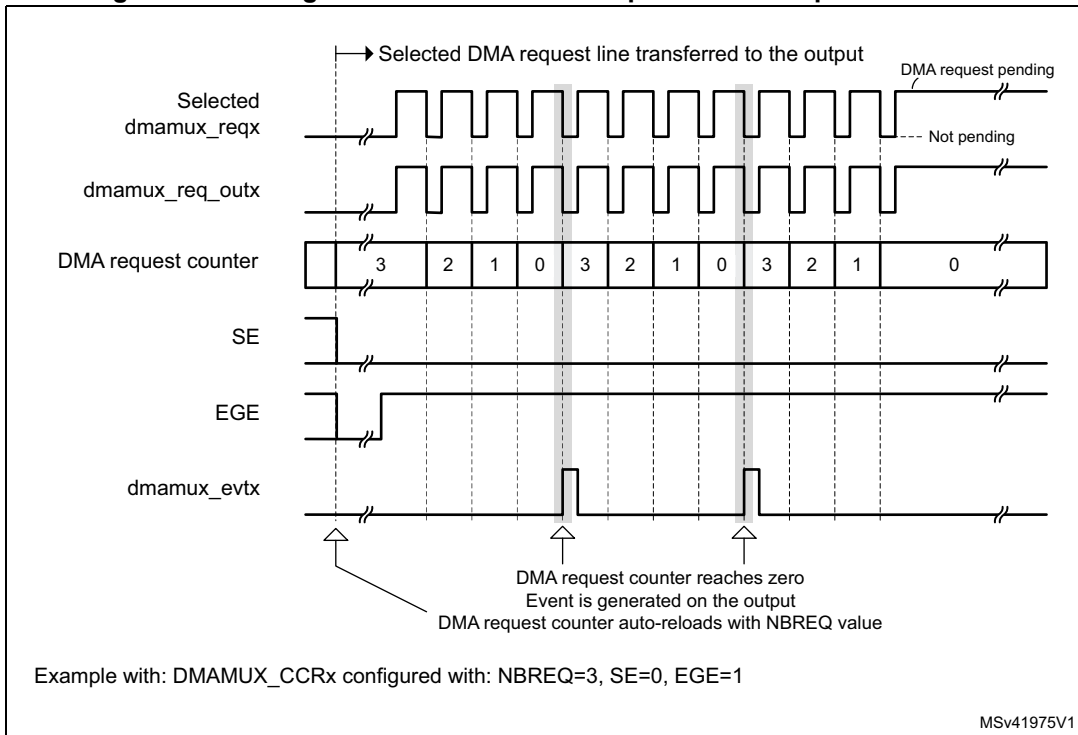


Figure 31. Event generation of the DMA request line multiplexer channel



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 30](#) and [Figure 31](#).

*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 while the value of the DMA request counter is lower than the programmed NBREQ field value of the multiplexer channel x, the synchronization overrun flag bit SOFx is set in the DMAMUX\_CSR status register.

*Note:* The request multiplexer channel x synchronization shall 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 will be 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 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 de-asserted), 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 will be 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 shall only be written by software when the enable GE bit of the corresponding generator channel x is disabled.

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 while the value of the DMAMUX request generator counter is lower than the programmed GNBREQ field value of the corresponding request generator channel x, 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 shall 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 will be 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 54. 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 shall be aligned with the data size.

### 12.6.1 DMAMUX request line multiplexer channel x configuration register (DMAMUX\_CxCR)

Address offset: 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		
			r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
						r/w	r/w		r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SYNC\_ID[4:0]**: Synchronization identification

Selects the synchronization input (see [Table 52: 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 edge

Bit 16 **SE**: Synchronization enable

0: synchronization disabled

1: synchronization enabled

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **EGE**: Event generation enable

0: event generation disabled

1: event generation enabled

Bit 8 **SOIE**: Synchronization overrun interrupt enable

0: interrupt disabled

1: interrupt enabled

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **DMAREQ\_ID[6:0]**: DMA request identification

Selects the input DMA request. See the DMAMUX table about assignments of multiplexer inputs to resources.

### 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 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.	CSOF13	CSOF12	CSOF11	CSOF10	CSOF9	CSOF8	CSOF7	CSOF6	CSOF5	CSOF4	CSOF3	CSOF2	CSOF1	CSOF0
		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-0) (x = 0 to 3)

Reset value: 0x0000 0000

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

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

Bits 23:19 **GNBREQ[4:0]**: Number of DMA requests to be generated (minus 1)

Defines the number of DMA requests to be generated after a trigger event. The actual number of generated DMA requests is GNBREQ+1.

*Note: This field shall only be written 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. none trigger detection nor generation.

01: rising edge

10: falling edge

11: rising and falling edge

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OF3	OF2	OF1	OF0
												r	r	r	r

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

Bits 3:0 **OF[3:0]**: Trigger overrun event flag

The flag is set when a trigger event occurs on DMA request generator channel x, while the DMA request generator counter value is lower than GNBREQ.

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COF3	COF2	COF1	COF0
												w	w	w	w

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

Bits 3:0 **COF[3:0]**: Clear trigger overrun event flag

Writing 1 in each bit clears the corresponding overrun flag OFx in the DMAMUX\_RGSR register.

### 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 55. DMAMUX register map and reset values**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
0x000	DMAMUX_C0CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]																			
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0												
0x004	DMAMUX_C1CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0	0												
0x008	DMAMUX_C2CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0												
0x00C	DMAMUX_C3CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0											
0x010	DMAMUX_C4CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0											
0x014	DMAMUX_C5CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0										
0x018	DMAMUX_C6CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0									
0x01C	DMAMUX_C7CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x020	DMAMUX_C8CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x024	DMAMUX_C9CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0x028	DMAMUX_C10CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x02C	DMAMUX_C11CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x030	DMAMUX_C12CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x034	DMAMUX_C13CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x038 - 0x07C	Reserved	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res				
0x080	DMAMUX_CSR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res				
	Reset value																																													
0x084	DMAMUX_CFR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res		
	Reset value																																													



Table 55. 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	Res	Res	
0x100	DMAMUX_RG0CR	Res	Res	Res	Res	Res	Res	Res	Res	GNBREQ[4:0]				GPOL	[1:0]	GE	Res	Res	Res	Res	Res	Res	Res	Res	OIE	Res	Res	Res	Res	SIG_ID[4:0]				
	Reset value									0	0	0	0	0	0	0	0	Res	Res	Res	Res	Res	Res	Res	Res	0	Res	Res	Res	Res	0	0	0	0
0x104	DMAMUX_RG1CR	Res	Res	Res	Res	Res	Res	Res	Res	GNBREQ[4:0]				GPOL	[1:0]	GE	Res	Res	Res	Res	Res	Res	Res	Res	OIE	Res	Res	Res	Res	SIG_ID[4:0]				
	Reset value									0	0	0	0	0	0	0	0	Res	Res	Res	Res	Res	Res	Res	Res	0	Res	Res	Res	Res	0	0	0	0
0x108	DMAMUX_RG2CR	Res	Res	Res	Res	Res	Res	Res	Res	GNBREQ[4:0]				GPOL	[1:0]	GE	Res	Res	Res	Res	Res	Res	Res	Res	OIE	Res	Res	Res	Res	SIG_ID[4:0]				
	Reset value									0	0	0	0	0	0	0	0	Res	Res	Res	Res	Res	Res	Res	Res	0	Res	Res	Res	Res	0	0	0	0
0x10C	DMAMUX_RG3CR	Res	Res	Res	Res	Res	Res	Res	Res	GNBREQ[4:0]				GPOL	[1:0]	GE	Res	Res	Res	Res	Res	Res	Res	Res	OIE	Res	Res	Res	Res	SIG_ID[4:0]				
	Reset value									0	0	0	0	0	0	0	0	Res	Res	Res	Res	Res	Res	Res	Res	0	Res	Res	Res	Res	0	0	0	0
0x110 - 0x13C	Reserved	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	
0x140	DMAMUX_RGSR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	
	Reset value																																	
0x144	DMAMUX_RGCFR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
	Reset value																																	
0x148 - 0x3FC	Reserved	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res



## 13 Chrom-ART Accelerator™ controller (DMA2D)

### 13.1 DMA2D introduction

The Chrom-ART Accelerator™ (DMA2D) is a specialized DMA dedicated to image manipulation. It can perform the following operations:

- Filling a part or the whole of a destination image with a specific color
- Copying a part or the whole of a source image into a part or the whole of a destination image
- Copying a part or the whole of a source image into a part or the whole of a destination image with a pixel format conversion
- Blending a part and/or two complete source images with different pixel format and copy the result into a part or the whole of a destination image with a different color format.

All the classical color coding schemes are supported from 4-bit up to 32-bit per pixel with indexed or direct color mode. The DMA2D has its own dedicated memories for CLUTs (color look-up tables).

### 13.2 DMA2D main features

The main DMA2D features are:

- Single AHB master bus architecture.
- AHB slave programming interface supporting 8/16/32-bit accesses (except for CLUT accesses which are 32-bit).
- User programmable working area size
- User programmable offset for sources and destination areas expressed in pixels or bytes
- User programmable sources and destination addresses on the whole memory space
- Up to 2 sources with blending operation
- Alpha value can be modified (source value, fixed value or modulated value)
- User programmable source and destination color format
- Up to 11 color formats supported from 4-bit up to 32-bit per pixel with indirect or direct color coding
- 2 internal memories for CLUT storage in indirect color mode
- Automatic CLUT loading or CLUT programming via the CPU
- User programmable CLUT size
- Internal timer to control AHB bandwidth
- 6 operating modes: register-to-memory, memory-to-memory, memory-to-memory with pixel format conversion, memory-to-memory with pixel format conversion and blending, memory-to-memory with pixel format conversion, blending and fixed color foreground, and memory-to-memory with pixel format conversion, blending and fixed color background.

- Area filling with a fixed color
- Copy from an area to another
- Copy with pixel format conversion between source and destination images
- Copy from two sources with independent color format and blending
- Output buffer byte swapping to support refresh of displays through parallel interface
- Abort and suspend of DMA2D operations
- Watermark interrupt on a user programmable destination line
- Interrupt generation on bus error or access conflict
- Interrupt generation on process completion

## 13.3 DMA2D functional description

### 13.3.1 General description

The DMA2D controller performs direct memory transfer. As an AHB master, it can take the control of the AHB bus matrix to initiate AHB transactions.

The DMA2D can operate in the following modes:

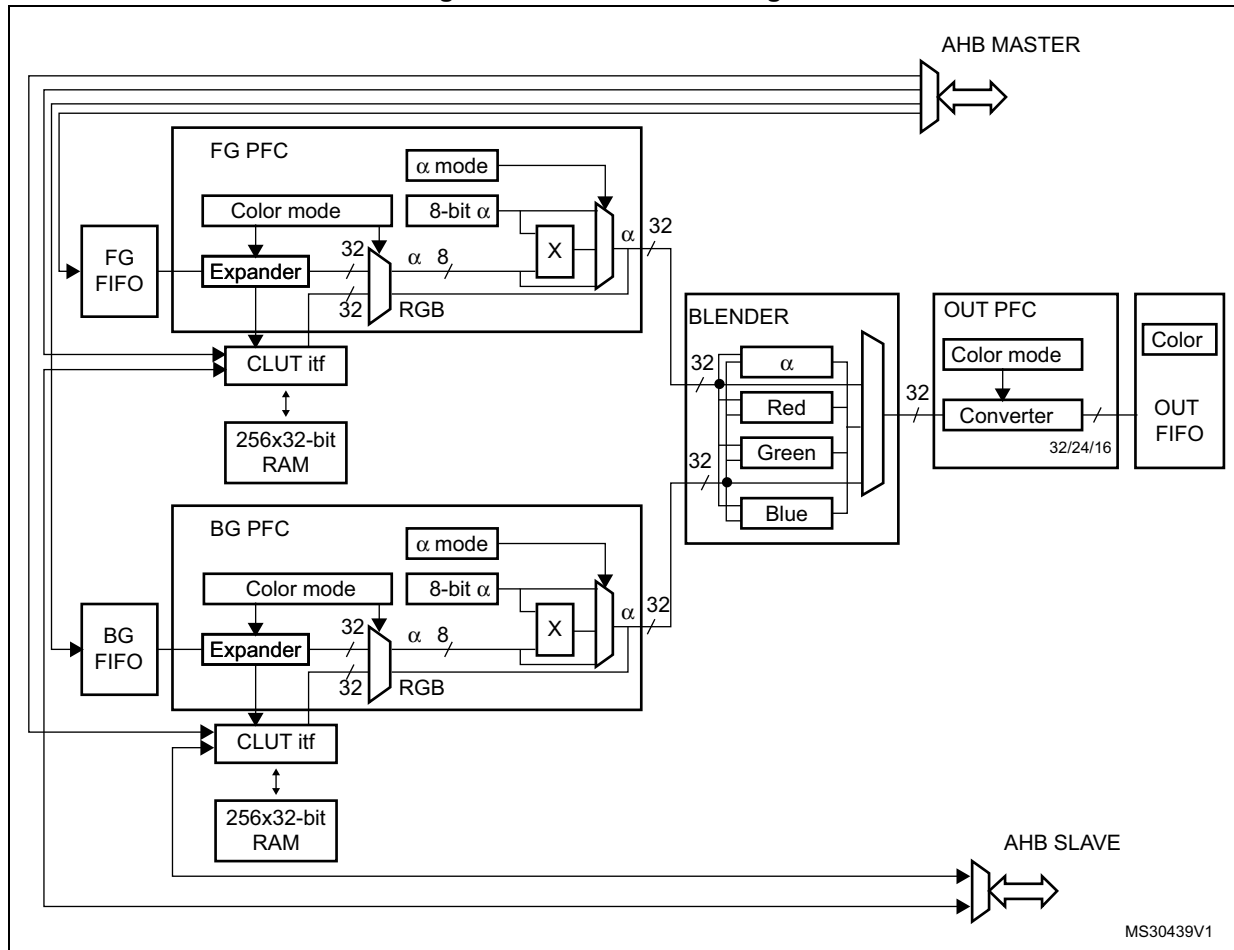
- Register-to-memory
- Memory-to-memory
- Memory-to-memory with Pixel Format Conversion
- Memory-to-memory with Pixel Format Conversion and Blending
- Memory-to memory with pixel format conversion, blending and fixed color foreground
- Memory-to memory with pixel format conversion, blending and fixed color background

The AHB slave port is used to program the DMA2D controller.

The block diagram of the DMA2D is shown in [Figure 32: DMA2D block diagram](#).



Figure 32. DMA2D block diagram



### 13.3.2 DMA2D control

The DMA2D controller is configured through the DMA2D Control Register (DMA2D\_CR) which allows selecting:

The user application can perform the following operations:

- Select the operating mode
- Enable/disable the DMA2D interrupt
- Start/suspend/abort ongoing data transfers

### 13.3.3 DMA2D foreground and background FIFOs

The DMA2D foreground (FG) FG FIFO and background (BG) FIFO fetch the input data to be copied and/or processed.

The FIFOs fetch the pixels according to the color format defined in their respective pixel format converter (PFC).

They are programmed through a set of control registers:

- DMA2D foreground memory address register (DMA2D\_FGMAR)
- DMA2D foreground offset register (DMA2D\_FGOR)
- DMA2D background memory address register (DMA2D\_BGMAR)
- DMA2D background offset register (DMA2D\_BGBOR)
- DMA2D number of lines register (number of lines and pixel per lines) (DMA2D\_NLR)

When the DMA2D operates in register-to-memory mode, none of the FIFOs is activated.

When the DMA2D operates in memory-to-memory mode (no pixel format conversion nor blending operation), only the FG FIFO is activated and acts as a buffer.

When the DMA2D operates in memory-to-memory operation with pixel format conversion (no blending operation), the BG FIFO is not activated.

### 13.3.4 DMA2D foreground and background pixel format converter (PFC)

DMA2D foreground pixel format converter (PFC) and background pixel format converter perform the pixel format conversion to generate a 32-bit per pixel value. The PFC can also modify the alpha channel.

The first stage of the converter converts the color format. The original color format of the foreground pixel and background pixels are configured through the CM[3:0] bits of the DMA2D\_FGPFCCR and DMA2D\_BGPFCCR, respectively.

The supported input formats are given in [Table 56: Supported color mode in input](#).

**Table 56. Supported color mode in input**

CM[3:0]	Color mode
0000	ARGB8888
0001	RGB888
0010	RGB565
0011	ARGB1555
0100	ARGB4444
0101	L8
0110	AL44
0111	AL88
1000	L4
1001	A8
1010	A4

The color format are coded as follows:

- Alpha value field: transparency  
0xFF value corresponds to an opaque pixel and 0x00 to a transparent one.
- R field for Red
- G field for Green
- B field for Blue
- L field: luminance  
This field is the index to a CLUT to retrieve the three/four RGB/ARGB components.

If the original format was direct color mode (ARGB/RGB), then the extension to 8-bit per channel is performed by copying the MSBs into the LSBs. This ensures a perfect linearity of the conversion.

If the original format is indirect color mode (L/AL), a CLUT is required and each pixel format converter is associated with a 256 entry 32-bit CLUT.

If the original format does not include an alpha channel, the alpha value is automatically set to 0xFF (opaque).

For the specific alpha mode A4 and A8, no color information is stored nor indexed. The color to be used for the image generation is fixed and is defined in the DMA2D\_FGCOLR for foreground pixels and in the DMA2D\_BGCOLR register for background pixels.

The order of the fields in the system memory is defined in [Table 57: Data order in memory](#).

**Table 57. Data order in memory**

Color Mode	@ + 3	@ + 2	@ + 1	@ + 0
ARGB8888	A <sub>0</sub> [7:0]	R <sub>0</sub> [7:0]	G <sub>0</sub> [7:0]	B <sub>0</sub> [7:0]
RGB888	B <sub>1</sub> [7:0]	R <sub>0</sub> [7:0]	G <sub>0</sub> [7:0]	B <sub>0</sub> [7:0]
	G <sub>2</sub> [7:0]	B <sub>2</sub> [7:0]	R <sub>1</sub> [7:0]	G <sub>1</sub> [7:0]
	R <sub>3</sub> [7:0]	G <sub>3</sub> [7:0]	B <sub>3</sub> [7:0]	R <sub>2</sub> [7:0]
RGB565	R <sub>1</sub> [4:0]G <sub>1</sub> [5:3]	G <sub>1</sub> [2:0]B <sub>1</sub> [4:0]	R <sub>0</sub> [4:0]G <sub>0</sub> [5:3]	G <sub>0</sub> [2:0]B <sub>0</sub> [4:0]
ARGB1555	A <sub>1</sub> [0]R <sub>1</sub> [4:0]G <sub>1</sub> [4:3]	G <sub>1</sub> [2:0]B <sub>1</sub> [4:0]	A <sub>0</sub> [0]R <sub>0</sub> [4:0]G <sub>0</sub> [4:3]	G <sub>0</sub> [2:0]B <sub>0</sub> [4:0]
ARGB4444	A <sub>1</sub> [3:0]R <sub>1</sub> [3:0]	G <sub>1</sub> [3:0]B <sub>1</sub> [3:0]	A <sub>0</sub> [3:0]R <sub>0</sub> [3:0]	G <sub>0</sub> [3:0]B <sub>0</sub> [3:0]
L8	L <sub>3</sub> [7:0]	L <sub>2</sub> [7:0]	L <sub>1</sub> [7:0]	L <sub>0</sub> [7:0]
AL44	A <sub>3</sub> [3:0]L <sub>3</sub> [3:0]	A <sub>2</sub> [3:0]L <sub>2</sub> [3:0]	A <sub>1</sub> [3:0]L <sub>1</sub> [3:0]	A <sub>0</sub> [3:0]L <sub>0</sub> [3:0]
AL88	A <sub>1</sub> [7:0]	L <sub>1</sub> [7:0]	A <sub>0</sub> [7:0]	L <sub>0</sub> [7:0]
L4	L <sub>7</sub> [3:0]L <sub>6</sub> [3:0]	L <sub>5</sub> [3:0]L <sub>4</sub> [3:0]	L <sub>3</sub> [3:0]L <sub>2</sub> [3:0]	L <sub>1</sub> [3:0]L <sub>0</sub> [3:0]
A8	A <sub>3</sub> [7:0]	A <sub>2</sub> [7:0]	A <sub>1</sub> [7:0]	A <sub>0</sub> [7:0]
A4	A <sub>7</sub> [3:0]A <sub>6</sub> [3:0]	A <sub>5</sub> [3:0]A <sub>4</sub> [3:0]	A <sub>3</sub> [3:0]A <sub>2</sub> [3:0]	A <sub>1</sub> [3:0]A <sub>0</sub> [3:0]

The 24-bit RGB888 aligned on 32-bit is supported through the ARGB8888 mode.

Once the 32-bit value is generated, the alpha channel can be modified according to the AM[1:0] field of the DMA2D\_FGPFCCR/DMA2D\_BGPFCCR registers as shown in [Table 58: Alpha mode configuration](#).

The alpha channel can be:

- kept as it is (no modification),
- replaced by the ALPHA[7:0] value of DMA2D\_FGPFCCR/DMA2D\_BGPFCCR,
- or replaced by the original alpha value multiplied by the ALPHA[7:0] value of DMA2D\_FGPFCCR/DMA2D\_BGPFCCR divided by 255.

**Table 58. Alpha mode configuration**

AM[1:0]	Alpha mode
00	No modification
01	Replaced by value in DMA2D_xxPFCCR
10	Replaced by original value multiplied by the value in DMA2D_xxPFCCR / 255
11	Reserved

*Note:* To support the alternate format, the incoming alpha value can be inverted setting the AI bit of the DMA2D\_FGPFCCR/DMA2D\_BGPFCCR registers. This applies also to the Alpha value stored in the DMA2D\_FGPFCCR/DMA2D\_BGPFCCR and in the CLUT.

The R and B fields can also be swapped setting the RBS bit of the DMA2D\_FGPFCCR/DMA2D\_BGPFCCR registers. This applies also to the RGB order used in the CLUT and in the DMA2D\_FGCOLR/DMA2D\_BGCOLR registers.

### 13.3.5 DMA2D foreground and background CLUT interface

The CLUT interface manages the CLUT memory access and the automatic loading of the CLUT.

Three kinds of accesses are possible:

- CLUT read by the PFC during pixel format conversion operation
- CLUT accessed through the AHB slave port when the CPU is reading or writing data into the CLUT
- CLUT written through the AHB master port when an automatic loading of the CLUT is performed

The CLUT memory loading can be done in two different ways:

- Automatic loading
  - The following sequence should be followed to load the CLUT:
    - a) Program the CLUT address into the DMA2D\_FGCMAR register (foreground CLUT) or DMA2D\_BGCMAR register (background CLUT)
    - b) Program the CLUT size in the CS[7:0] field of the DMA2D\_FGPFCCR register (foreground CLUT) or DMA2D\_BGPFCCR register (background CLUT).
    - c) Set the START bit of the DMA2D\_FGPFCCR register (foreground CLUT) or DMA2D\_BGPFCCR register (background CLUT) to start the transfer. During this automatic loading process, the CLUT is not accessible by the CPU. If a conflict

occurs, a CLUT access error interrupt is raised assuming CAEIE is set to '1' in DMA2D\_CR.

- Manual loading  
The application has to program the CLUT manually through the DMA2D AHB slave port to which the local CLUT memory is mapped. The foreground CLUT is located at address offset 0x0400 and the background CLUT at address offset 0x0800.

The CLUT format can be 24 or 32 bits. It is configured through the CCM bit of the DMA2D\_FGPFCCR register (foreground CLUT) or DMA2D\_BGPFCCR register (background CLUT) as shown in [Table 59: Supported CLUT color mode](#).

**Table 59. Supported CLUT color mode**

CCM	CLUT color mode
0	32-bit ARGB8888
1	24-bit RGB888

The way the CLUT data are organized in the system memory is specified in [Table 60: CLUT data order in system memory](#).

**Table 60. CLUT data order in system memory**

CLUT Color Mode	@ + 3	@ + 2	@ + 1	@ + 0
ARGB8888	A <sub>0</sub> [7:0]	R <sub>0</sub> [7:0]	G <sub>0</sub> [7:0]	B <sub>0</sub> [7:0]
RGB888	B <sub>1</sub> [7:0]	R <sub>0</sub> [7:0]	G <sub>0</sub> [7:0]	B <sub>0</sub> [7:0]
	G <sub>2</sub> [7:0]	B <sub>2</sub> [7:0]	R <sub>1</sub> [7:0]	G <sub>1</sub> [7:0]
	R <sub>3</sub> [7:0]	G <sub>3</sub> [7:0]	B <sub>3</sub> [7:0]	R <sub>2</sub> [7:0]

### 13.3.6 DMA2D blender

The DMA2D blender blends the source pixels by pair to compute the resulting pixel.

The blending is performed according to the following equation:

$$\text{with } \alpha_{\text{Mult}} = \frac{\alpha_{\text{FG}} \cdot \alpha_{\text{BG}}}{255}$$

$$\alpha_{\text{OUT}} = \alpha_{\text{FG}} + \alpha_{\text{BG}} - \alpha_{\text{Mult}}$$

$$C_{\text{OUT}} = \frac{C_{\text{FG}} \cdot \alpha_{\text{FG}} + C_{\text{BG}} \cdot \alpha_{\text{BG}} - C_{\text{BG}} \cdot \alpha_{\text{Mult}}}{\alpha_{\text{OUT}}} \quad \text{with } C = R \text{ or } G \text{ or } B$$

*Division is rounded to the nearest lower integer*

No configuration register is required by the blender. The blender usage depends on the DMA2D operating mode defined in MODE[2:0] field of the DMA2D\_CR register.

### 13.3.7 DMA2D output PFC

The output PFC performs the pixel format conversion from 32 bits to the output format defined in the CM[2:0] field of the DMA2D output pixel format converter configuration register (DMA2D\_OPFCCR).

The supported output formats are given in [Table 61: Supported color mode in output](#)

**Table 61. Supported color mode in output**

CM[2:0]	Color mode
000	ARGB8888
001	RGB888
010	RGB565
011	ARGB1555
100	ARGB4444

*Note:* To support the alternate format, the calculated alpha value can be inverted setting the AI bit of the DMA2D\_OPFCCR registers. This applies also to the Alpha value used in the DMA2D\_OCOLR.

The R and B fields can also be swapped setting the RBS bit of the DMA2D\_OPFCCR registers. This applies also to the RGB order used in the DMA2D\_OCOLR.

### 13.3.8 DMA2D output FIFO

The output FIFO programs the pixels according to the color format defined in the output PFC.

The destination area is defined through a set of control registers:

- DMA2D output memory address register (DMA2D\_OMAR)
- DMA2D output offset register (DMA2D\_OOR)
- DMA2D number of lines register (number of lines and pixel per lines) (DMA2D\_NLR)

If the DMA2D operates in register-to-memory mode, the configured output rectangle is filled by the color specified in the DMA2D output color register (DMA2D\_OCOLR) which contains a fixed 32-bit, 24-bit or 16-bit value. The format is selected by the CM[2:0] field of the DMA2D\_OPFCCR register.

The data are stored into the memory in the order defined in [Table 62: Data order in memory](#)

**Table 62. Data order in memory**

Color Mode	@ + 3	@ + 2	@ + 1	@ + 0
ARGB8888	A <sub>0</sub> [7:0]	R <sub>0</sub> [7:0]	G <sub>0</sub> [7:0]	B <sub>0</sub> [7:0]
RGB888	B <sub>1</sub> [7:0]	R <sub>0</sub> [7:0]	G <sub>0</sub> [7:0]	B <sub>0</sub> [7:0]
	G <sub>2</sub> [7:0]	B <sub>2</sub> [7:0]	R <sub>1</sub> [7:0]	G <sub>1</sub> [7:0]
	R <sub>3</sub> [7:0]	G <sub>3</sub> [7:0]	B <sub>3</sub> [7:0]	R <sub>2</sub> [7:0]
RGB565	R <sub>1</sub> [4:0]G <sub>1</sub> [5:3]	G <sub>1</sub> [2:0]B <sub>1</sub> [4:0]	R <sub>0</sub> [4:0]G <sub>0</sub> [5:3]	G <sub>0</sub> [2:0]B <sub>0</sub> [4:0]

**Table 62. Data order in memory (continued)**

Color Mode	@ + 3	@ + 2	@ + 1	@ + 0
ARGB1555	A <sub>1</sub> [0]R <sub>1</sub> [4:0]G <sub>1</sub> [4:3]	G <sub>1</sub> [2:0]B <sub>1</sub> [4:0]	A <sub>0</sub> [0]R <sub>0</sub> [4:0]G <sub>0</sub> [4:3]	G <sub>0</sub> [2:0]B <sub>0</sub> [4:0]
ARGB4444	A <sub>1</sub> [3:0]R <sub>1</sub> [3:0]	G <sub>1</sub> [3:0]B <sub>1</sub> [3:0]	A <sub>0</sub> [3:0]R <sub>0</sub> [3:0]	G <sub>0</sub> [3:0]B <sub>0</sub> [3:0]

The RGB888 aligned on 32-bit is supported through the ARGB8888 mode.

### 13.3.9 DMA2D output FIFO byte reordering

The output FIFO bytes can be reordered to support display frame buffer update through a parallel interface (F(S)MC) directly from the DMA2D.

The reordering of bytes can be done using:

- RBS bit to swap Red and Blue component
- SB bit to swap byte two by two in the output FIFO

When the byte swapping is activated (SB field of the DMA2D\_OPFCR is set), the number of pixel per line (PL field of the DMA2D\_NLR) must be even and the output memory address (MA field of the DMA2D\_OMAR) must be even, and the output line offset computed in bytes (resulting from LOM field of DMA2D\_CR and LO field of DMA2D\_OOR values) must be even. If not a configuration error will be detected.

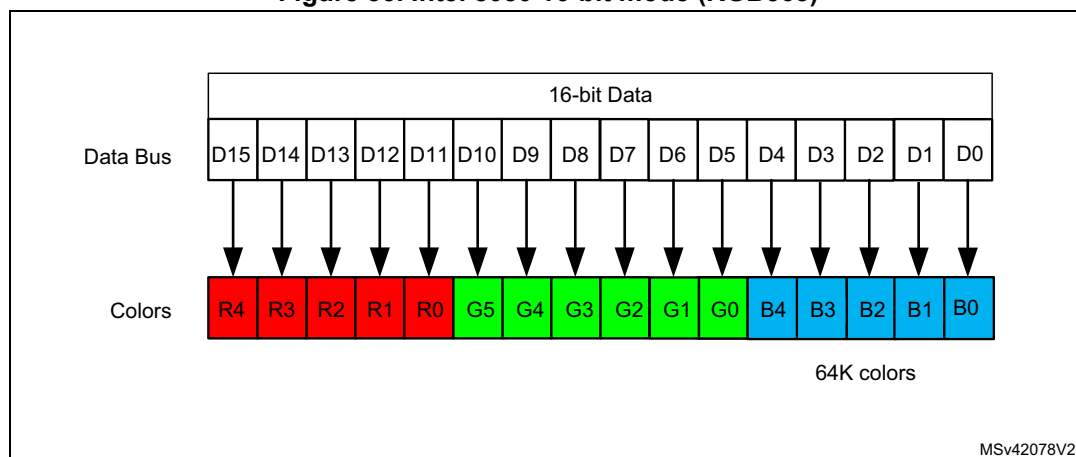
**Table 63. Standard data order in memory**

Color Mode	@ + 3	@ + 2	@ + 1	@ + 0
RGB888	B <sub>1</sub> [7:0]	R <sub>0</sub> [7:0]	G <sub>0</sub> [7:0]	B <sub>0</sub> [7:0]
	G <sub>2</sub> [7:0]	B <sub>2</sub> [7:0]	R <sub>1</sub> [7:0]	G <sub>1</sub> [7:0]
	R <sub>3</sub> [7:0]	G <sub>3</sub> [7:0]	B <sub>3</sub> [7:0]	R <sub>2</sub> [7:0]
RGB565	R <sub>1</sub> [4:0]G <sub>1</sub> [5:3]	G <sub>1</sub> [2:0]B <sub>1</sub> [4:0]	R <sub>0</sub> [4:0]G <sub>0</sub> [5:3]	G <sub>0</sub> [2:0]B <sub>0</sub> [4:0]

### 16-bit mode (RGB565)

This mode is supported without byte reordering by the DMA2D.

**Figure 33. Intel 8080 16-bit mode (RGB565)**

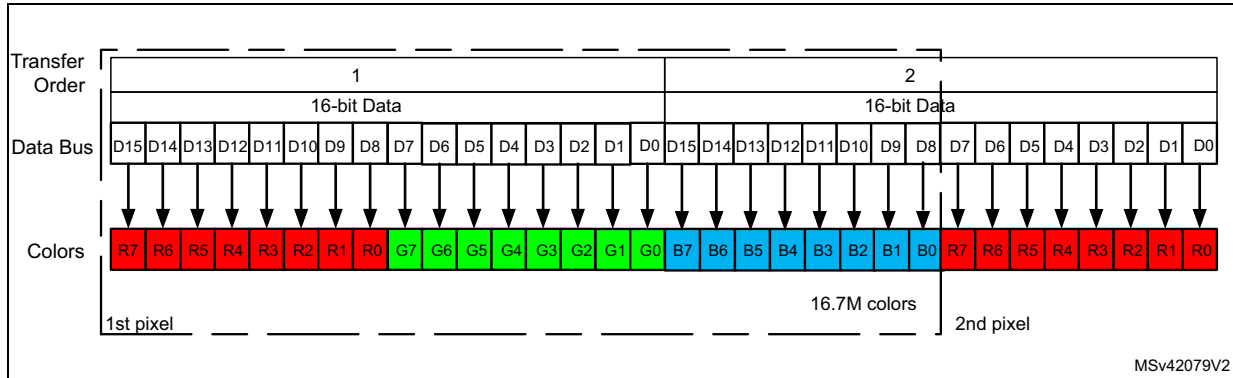


**18/24-bit mode (RGB888)**

This mode needs data reordering.

1. The Red and the Blue have to be swapped (setting the RBS bit)
2. The MSB and the LSB bytes of an half-word as to be swapped (setting the SB bit)

**Figure 34. Intel 8080 18/24-bit mode (RGB888)**



**Table 64. Output FIFO byte reordering steps**

Steps	@ + 3	@ + 2	@ + 1	@ + 0
Original data ordering	B <sub>1</sub> [7:0]	R <sub>0</sub> [7:0]	G <sub>0</sub> [7:0]	B <sub>0</sub> [7:0]
	G <sub>2</sub> [7:0]	B <sub>2</sub> [7:0]	R <sub>1</sub> [7:0]	G <sub>1</sub> [7:0]
	R <sub>3</sub> [7:0]	G <sub>3</sub> [7:0]	B <sub>3</sub> [7:0]	R <sub>2</sub> [7:0]
Setting the RBS bit				
Data ordering after red and blue swap (RBS set)	R <sub>1</sub> [7:0]	B <sub>0</sub> [7:0]	G <sub>0</sub> [7:0]	R <sub>0</sub> [7:0]
	G <sub>2</sub> [7:0]	R <sub>2</sub> [7:0]	B <sub>1</sub> [7:0]	G <sub>1</sub> [7:0]
	B <sub>3</sub> [7:0]	G <sub>3</sub> [7:0]	R <sub>3</sub> [7:0]	B <sub>2</sub> [7:0]
Setting the SB bit				
Data ordering after byte swapping (SB set)	B <sub>0</sub> [7:0]	R <sub>1</sub> [7:0]	R <sub>0</sub> [7:0]	G <sub>0</sub> [7:0]
	R <sub>2</sub> [7:0]	G <sub>2</sub> [7:0]	G <sub>1</sub> [7:0]	B <sub>1</sub> [7:0]
	G <sub>3</sub> [7:0]	B <sub>3</sub> [7:0]	B <sub>2</sub> [7:0]	R <sub>3</sub> [7:0]

**13.3.10 DMA2D AHB master port timer**

An 8-bit timer is embedded into the AHB master port to provide an optional limitation of the bandwidth on the crossbar.

This timer is clocked by the AHB clock and counts a dead time between two consecutive accesses. This limits the bandwidth usage.

The timer enabling and the dead time value are configured through the AHB master port timer configuration register (DMA2D\_AMPTCR).



### 13.3.11 DMA2D transactions

DMA2D transactions consist of a sequence of a given number of data transfers. The number of data and the width can be programmed by software.

Each DMA2D data transfer is composed of up to 4 steps:

1. Data loading from the memory location pointed by the DMA2D\_FGMAR register and pixel format conversion as defined in DMA2D\_FGCR.
2. Data loading from a memory location pointed by the DMA2D\_BGMAR register and pixel format conversion as defined in DMA2D\_BGCR.
3. Blending of all retrieved pixels according to the alpha channels resulting of the PFC operation on alpha values.
4. Pixel format conversion of the resulting pixels according to the DMA2D\_OCR register and programming of the data to the memory location addressed through the DMA2D\_OMAR register.

### 13.3.12 DMA2D configuration

Both source and destination data transfers can target peripherals and memories in the whole 4 Gbyte memory area, at addresses ranging between 0x0000 0000 and 0xFFFF FFFF.

The DMA2D can operate in any of the four following modes selected through MODE[2:0] bits of the DMA2D\_CR register:

- Register-to-memory
- Memory-to-memory
- Memory-to-memory with PFC
- Memory-to-memory with PFC and blending
- Memory-to-memory with PFC, blending and fixed FG color
- Memory-to-memory with PFC, blending and fixed BG color

#### Register-to-memory

The register-to-memory mode is used to fill a user defined area with a predefined color.

The color format is set in the DMA2D\_OPFCCR.

The DMA2D does not perform any data fetching from any source. It just writes the color defined in the DMA2D\_OCOLR register to the area located at the address pointed by the DMA2D\_OMAR and defined in the DMA2D\_NLR and DMA2D\_OOR.

#### Memory-to-memory

In memory-to-memory mode, the DMA2D does not perform any graphical data transformation. The foreground input FIFO acts as a buffer and the data are transferred from the source memory location defined in DMA2D\_FGMAR to the destination memory location pointed by DMA2D\_OMAR.

The color mode programmed in the CM[3:0] bits of the DMA2D\_FGPFCCR register defines the number of bits per pixel for both input and output.

The size of the area to be transferred is defined by the DMA2D\_NLR and DMA2D\_FGOR registers for the source, and by DMA2D\_NLR and DMA2D\_OOR registers for the destination.

## Memory-to-memory with PFC

In this mode, the DMA2D performs a pixel format conversion of the source data and stores them in the destination memory location.

The size of the areas to be transferred are defined by the DMA2D\_NLR and DMA2D\_FGOR registers for the source, and by DMA2D\_NLR and DMA2D\_OOR registers for the destination.

Data are fetched from the location defined in the DMA2D\_FGMAR register and processed by the foreground PFC. The original pixel format is configured through the DMA2D\_FGPFCCR register.

If the original pixel format is direct color mode, then the color channels are all expanded to 8 bits.

If the pixel format is indirect color mode, the associated CLUT has to be loaded into the CLUT memory.

The CLUT loading can be done automatically by following the sequence below:

1. Set the CLUT address into the DMA2D\_FGCMAR.
2. Set the CLUT size in the CS[7:0] bits of the DMA2D\_FGPFCCR register.
3. Set the CLUT format (24 or 32 bits) in the CCM bit of the DMA2D\_FGPFCCR register.
4. Start the CLUT loading by setting the START bit of the DMA2D\_FGPFCCR register.

Once the CLUT loading is complete, the CTCIF flag of the DMA2D\_IFR register is raised, and an interrupt is generated if the CTCIE bit is set in DMA2D\_CR. The automatic CLUT loading process can not work in parallel with classical DMA2D transfers.

The CLUT can also be filled by the CPU or by any other master through the APB port. The access to the CLUT is not possible when a DMA2D transfer is ongoing and uses the CLUT (indirect color format).

In parallel to the color conversion process, the alpha value can be added or changed depending on the value programmed in the DMA2D\_FGPFCCR register. If the original image does not have an alpha channel, a default alpha value of 0xFF is automatically added to obtain a fully opaque pixel. The alpha value can be modified according to the AM[1:0] bits of the DMA2D\_FGPFCCR register:

- It can be unchanged.
- It can be replaced by the value defined in the ALPHA[7:0] value of the DMA2D\_FGPFCCR register.
- It can be replaced by the original value multiplied by the ALPHA[7:0] value of the DMA2D\_FGPFCCR register divided by 255.

The resulting 32-bit data are encoded by the OUT PFC into the format specified by the CM[2:0] field of the DMA2D\_OPFCCR register. The output pixel format cannot be the indirect mode since no CLUT generation process is supported.

The processed data are written into the destination memory location pointed by DMA2D\_OMAR.

## Memory-to-memory with PFC and blending

In this mode, 2 sources are fetched in the foreground FIFO and background FIFO from the memory locations defined by DMA2D\_FGMAR and DMA2D\_BGMAR.

The two pixel format converters have to be configured as described in the memory-to-memory mode. Their configurations can be different as each pixel format converter are independent and have their own CLUT memory.

Once each pixel has been converted into 32 bits by their respective PFCs, they are blended according to the equation below:

$$\text{with } \alpha_{\text{Mult}} = \frac{\alpha_{\text{FG}} \cdot \alpha_{\text{BG}}}{255}$$

$$\alpha_{\text{OUT}} = \alpha_{\text{FG}} + \alpha_{\text{BG}} - \alpha_{\text{Mult}}$$

$$C_{\text{OUT}} = \frac{C_{\text{FG}} \cdot \alpha_{\text{FG}} + C_{\text{BG}} \cdot \alpha_{\text{BG}} - C_{\text{BG}} \cdot \alpha_{\text{Mult}}}{\alpha_{\text{OUT}}} \quad \text{with } C = R \text{ or } G \text{ or } B$$

*Division are rounded to the nearest lower integer*

The resulting 32-bit pixel value is encoded by the output PFC according to the specified output format, and the data are written into the destination memory location pointed by DMA2D\_OMAR.

### Memory-to-memory with PFC, blending and fixed color FG

In this mode, only 1 source is fetched in the background FIFO from the memory location defined by DMA2D\_BGMAR.

The value of the foreground color is given by the DMA2D\_FGCOLR register and the alpha value is set to 0xFF (opaque).

The alpha value can be replaced or modified according to the AM[1:0] and ALPHA[7:0] fields of the DMA2D\_FGPFCR.

The two pixel format converters have to be configured as described in the memory-to-memory mode. Their configurations can be different as each pixel format converter are independent and have their own CLUT memory

Once each pixel has been converted into 32 bits by their respective PFCs, they are blended together, and the resulting 32-bit pixel value is encoded by the output PFC according to the specified output format, and the data are written into the destination memory location pointed by DMA2D\_OMAR.

### Memory-to-memory with PFC, blending and fixed color BG

In this mode, only 1 source is fetched in the foreground FIFO from the memory location defined by DMA2D\_FGMAR.

The value of the background color is given by the DMA2D\_BGCOLR register and the alpha value is set to 0xFF (opaque).

The alpha value can be replaced or modified according to the AM[1:0] and ALPHA[7:0] fields of the DMA2D\_BGPFCR.

The two pixel format converters have to be configured as described in the memory-to-memory mode. Their configurations can be different as each pixel format converter are independent and have their own CLUT memory

Once each pixel has been converted into 32 bits by their respective PFCs, they are blended together, and the resulting 32-bit pixel value is encoded by the output PFC according to the specified output format, and the data are written into the destination memory location pointed by DMA2D\_OMAR.

### Configuration error detection

The DMA2D checks that the configuration is correct before any transfer. The configuration error interrupt flag is set by hardware when a wrong configuration is detected when a new transfer/automatic loading starts. An interrupt is then generated if the CEIE bit of the DMA2D\_CR is set.

The wrong configurations that can be detected are listed below:

- Foreground CLUT automatic loading: MA bits of DMA2D\_FGCMAR are not aligned with CCM of DMA2D\_FGPFCCR.
- Background CLUT automatic loading: MA bits of DMA2D\_BGCMAR are not aligned with CCM of DMA2D\_BGPFCCR
- Memory transfer (except in register-to-memory mode and except in memory-to-memory mode with blending and fixed color FG): MA bits of DMA2D\_FGMAR are not aligned with CM of DMA2D\_FGPFCCR
- Memory transfer (except in register-to-memory mode and except in memory-to-memory mode with blending and fixed color FG): CM bits of DMA2D\_FGPFCCR are invalid
- Memory transfer (except in register-to-memory mode and except in memory-to-memory mode with blending and fixed color FG): PL bits of DMA2D\_NLR are odd while CM of DMA2D\_FGPFCCR is A4 or L4
- Memory transfer (except in register-to-memory mode and except in memory-to-memory mode with blending and fixed color FG): LO bits of DMA2D\_FGOR are odd while CM of DMA2D\_FGPFCCR is A4 or L4 and LOM bit of the DMA2D\_CR is pixel mode
- Memory transfer (only in blending mode and except in memory-to-memory mode with blending and fixed color FG): MA bits of DMA2D\_BGMAR are not aligned with the CM of DMA2D\_BGPFCCR
- Memory transfer: (only in blending mode and in blending with fixed color FG mode) CM bits of DMA2D\_BGPFCCR are invalid
- Memory transfer (only in blending mode and in blending with fixed color FG mode): PL bits of DMA2D\_NLR odd while CM of DMA2D\_BGPFCCR is A4 or L4
- Memory transfer (only in blending mode and in blending with fixed color FG mode): LO bits of DMA2D\_BGOR are odd while CM of DMA2D\_BGPFCCR is A4 or L4 and LOM bit of the DMA2D\_CR is pixel mode
- Memory transfer (except in memory to memory mode): MA bits of DMA2D\_OMAR are not aligned with CM bits of DMA2D\_OPFCCR.
- Memory transfer (except in memory to memory mode): CM bits of DMA2D\_OPFCCR are invalid
- Memory transfer with byte swapping: PL bits of DMA2D\_NLR are odd or MA bits of the DMA2D\_OMAR are odd or LO in bytes (resulting from LOM bit of the DMA2D\_CR and LO bits of DMA2D\_OOR values) are odd while SB bit of DMA2D\_OPFCCR is set
- Memory transfer: NL bits of DMA2D\_NLR = 0

- Memory transfer: PL bits of DMA2D\_NLR = 0
- Memory transfer: MODE bits of DMA2D\_CR are invalid

### 13.3.13 DMA2D transfer control (start, suspend, abort and completion)

Once the DMA2D is configured, the transfer can be launched by setting the START bit of the DMA2D\_CR register. Once the transfer is completed, the START bit is automatically reset and the TCIF flag of the DMA2D\_ISR register is raised. An interrupt can be generated if the TCIE bit of the DMA2D\_CR is set.

The user application can suspend the DMA2D at any time by setting the SUSP bit of the DMA2D\_CR register. The transaction can then be aborted by setting the ABORT bit of the DMA2D\_CR register or can be restarted by resetting the SUSP bit of the DMA2D\_CR register.

The user application can abort at any time an ongoing transaction by setting the ABORT bit of the DMA2D\_CR register. In this case, the TCIF flag is not raised.

Automatic CLUT transfers can also be aborted or suspended by using the ABORT or the SUSP bit of the DMA2D\_CR register.

### 13.3.14 Watermark

A watermark can be programmed to generate an interrupt when the last pixel of a given line has been written to the destination memory area.

The line number is defined in the LW[15:0] field of the DMA2D\_LWR register.

When the last pixel of this line has been transferred, the TWIF flag of the DMA2D\_ISR register is raised and an interrupt is generated if the TWIE bit of the DMA2D\_CR is set.

### 13.3.15 Error management

Two kind of errors can be triggered:

- AHB master port errors signaled by the TEIF flag of the DMA2D\_ISR register.
- Conflicts caused by CLUT access (CPU trying to access the CLUT while a CLUT loading or a DMA2D transfer is ongoing) signalled by the CAEIF flag of the DMA2D\_ISR register.

Both flags are associated to their own interrupt enable flag in the DMA2D\_CR register to generate an interrupt if need be (TEIE and CAEIE).

### 13.3.16 AHB dead time

To limit the AHB bandwidth usage, a dead time between two consecutive AHB accesses can be programmed.

This feature can be enabled by setting the EN bit in the DMA2D\_AMTCR register.

The dead time value is stored in the DT[7:0] field of the DMA2D\_AMTCR register. This value represents the guaranteed minimum number of cycles between two consecutive transactions on the AHB bus.

The update of the dead time value while the DMA2D is running will be taken into account for the next AHB transfer.

## 13.4 DMA2D interrupts

An interrupt can be generated on the following events:

- Configuration error
- CLUT transfer complete
- CLUT access error
- Transfer watermark reached
- Transfer complete
- Transfer error

Separate interrupt enable bits are available for flexibility.

**Table 65. DMA2D interrupt requests**

Interrupt event	Event flag	Enable control bit
Configuration error	CEIF	CEIE
CLUT transfer complete	CTCIF	CTCIE
CLUT access error	CAEIF	CAEIE
Transfer watermark	TWF	TWIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE

## 13.5 DMA2D registers

### 13.5.1 DMA2D control register (DMA2D\_CR)

Address offset: 0x0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CEIE	CTCIE	CAEIE	TWIE	TCIE	TEIE	Res.	LOM	Res.	Res.	Res.	ABORT	SUSP	START
		rw	rw	rw	rw	rw	rw		rw				rs	rw	rs

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

Bits 18:16 **MODE[2:0]**: DMA2D mode

These bits are set and cleared by software. They cannot be modified while a transfer is ongoing.

000: Memory-to-memory (FG fetch only)

001: Memory-to-memory with PFC (FG fetch only with FG PFC active)

010: Memory-to-memory with blending (FG and BG fetch with PFC and blending)

011: Register-to-memory (no FG nor BG, only output stage active)

100: Memory-to-memory with Blending and fixed color FG (BG fetch only with FG and BG PFC active)

101: Memory-to-memory with Blending and fixed color BG (BG fetch only with FG and BG PFC active)

others: meaningless

Bits 15:14 Reserved, must be kept at reset value

Bit 13 **CEIE**: Configuration Error Interrupt Enable

This bit is set and cleared by software.

0: CE interrupt disable

1: CE interrupt enable

Bit 12 **CTCIE**: CLUT transfer complete interrupt enable

This bit is set and cleared by software.

0: CTC interrupt disable

1: CTC interrupt enable

Bit 11 **CAEIE**: CLUT access error interrupt enable

This bit is set and cleared by software.

0: CAE interrupt disable

1: CAE interrupt enable

Bit 10 **TWIE**: Transfer watermark interrupt enable

This bit is set and cleared by software.

0: TW interrupt disable

1: TW interrupt enable

- Bit 9 **TCIE**: Transfer complete interrupt enable  
This bit is set and cleared by software.  
0: TC interrupt disable  
1: TC interrupt enable
- Bit 8 **TEIE**: Transfer error interrupt enable  
This bit is set and cleared by software.  
0: TE interrupt disable  
1: TE interrupt enable
- Bit 7 Reserved, must be kept at reset value
- Bit 6 **LOM**: Line Offset Mode  
This bit configures how is expressed the line offset (pixels or bytes) for the foreground, background and output.  
This bit is set and cleared by software. It can not be modified while a transfer is on going.  
0: Line offsets are expressed in pixels  
1: Line offsets are expressed in bytes
- Bits 5:3 Reserved, must be kept at reset value
- Bit 2 **ABORT**: Abort  
This bit can be used to abort the current transfer. This bit is set by software and is automatically reset by hardware when the START bit is reset.  
0: No transfer abort requested  
1: Transfer abort requested
- Bit 1 **SUSP**: Suspend  
This bit can be used to suspend the current transfer. This bit is set and reset by software. It is automatically reset by hardware when the START bit is reset.  
0: Transfer not suspended  
1: Transfer suspended
- Bit 0 **START**: Start  
This bit can be used to launch the DMA2D according to the parameters loaded in the various configuration registers. This bit is automatically reset by the following events:
- At the end of the transfer
  - When the data transfer is aborted by the user application by setting the ABORT bit in DMA2D\_CR
  - When a data transfer error occurs
  - When the data transfer has not started due to a configuration error or another transfer operation already ongoing (automatic CLUT loading).



### 13.5.2 DMA2D Interrupt Status Register (DMA2D\_ISR)

Address offset: 0x0004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	CEIF	CTCIF	CAEIF	TWIF	TCIF	TEIF
										r	r	r	r	r	r

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

Bit 5 **CEIF**: Configuration error interrupt flag

This bit is set when the START bit of DMA2D\_CR, DMA2DFGPFCCR or DMA2D\_BGPFCCR is set and a wrong configuration has been programmed.

Bit 4 **CTCIF**: CLUT transfer complete interrupt flag

This bit is set when the CLUT copy from a system memory area to the internal DMA2D memory is complete.

Bit 3 **CAEIF**: CLUT access error interrupt flag

This bit is set when the CPU accesses the CLUT while the CLUT is being automatically copied from a system memory to the internal DMA2D.

Bit 2 **TWIF**: Transfer watermark interrupt flag

This bit is set when the last pixel of the watermarked line has been transferred.

Bit 1 **TCIF**: Transfer complete interrupt flag

This bit is set when a DMA2D transfer operation is complete (data transfer only).

Bit 0 **TEIF**: Transfer error interrupt flag

This bit is set when an error occurs during a DMA transfer (data transfer or automatic CLUT loading).

### 13.5.3 DMA2D interrupt flag clear register (DMA2D\_IFCR)

Address offset: 0x0008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	CCEIF	CCTCIF	CAECIF	CTWIF	CTCIF	CTEIF
										rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

- Bit 5 **CCEIF**: Clear configuration error interrupt flag  
Programming this bit to 1 clears the CEIF flag in the DMA2D\_ISR register
- Bit 4 **CCTCIF**: Clear CLUT transfer complete interrupt flag  
Programming this bit to 1 clears the CTCIF flag in the DMA2D\_ISR register
- Bit 3 **CAECIF**: Clear CLUT access error interrupt flag  
Programming this bit to 1 clears the CAEIF flag in the DMA2D\_ISR register
- Bit 2 **CTWIF**: Clear transfer watermark interrupt flag  
Programming this bit to 1 clears the TWIF flag in the DMA2D\_ISR register
- Bit 1 **CTCIF**: Clear transfer complete interrupt flag  
Programming this bit to 1 clears the TCIF flag in the DMA2D\_ISR register
- Bit 0 **CTEIF**: Clear Transfer error interrupt flag  
Programming this bit to 1 clears the TEIF flag in the DMA2D\_ISR register

### 13.5.4 DMA2D foreground memory address register (DMA2D\_FGMAR)

Address offset: 0x000C

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]**: Memory address

Address of the data used for the foreground image. This register can only be written when data transfers are disabled. Once the data transfer has started, this register is read-only.

The address alignment must match the image format selected e.g. a 32-bit per pixel format must be 32-bit aligned, a 16-bit per pixel format must be 16-bit aligned and a 4-bit per pixel format must be 8-bit aligned.

### 13.5.5 DMA2D foreground offset register (DMA2D\_FGOR)

Address offset: 0x0010

Reset value: 0x0000 0000

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

The line offset used for the foreground image, expressed in pixel when the LOM bit is reset and in byte when the LOM bit is set.

When expressed in pixels, only LO[13:0] is considered, LO[15:14] are ignored.

This value is used for the address generation. It is added at the end of each line to determine the starting address of the next line.

These bits can only be written when data transfers are disabled. Once data transfer has started, they become read-only.

If the image format is 4-bit per pixel, the line offset must be even.

### 13.5.6 DMA2D background memory address register (DMA2D\_BGMAR)

Address offset: 0x0014

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]**: Memory address

Address of the data used for the background image. This register can only be written when data transfers are disabled. Once a data transfer has started, this register is read-only.

The address alignment must match the image format selected e.g. a 32-bit per pixel format must be 32-bit aligned, a 16-bit per pixel format must be 16-bit aligned and a 4-bit per pixel format must be 8-bit aligned.

### 13.5.7 DMA2D background offset register (DMA2D\_BGOR)

Address offset: 0x0018

Reset value: 0x0000 0000

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

The line offset used for the background image, expressed in pixel when the LOM bit is reset and in byte when the LOM bit is set.

When expressed in pixels, only LO[13:0] is considered, LO[15:14] are ignored.

This value is used for the address generation. It is added at the end of each line to determine the starting address of the next line.

These bits can only be written when data transfers are disabled. Once data transfer has started, they become read-only.

If the image format is 4-bit per pixel, the line offset must be even.

### 13.5.8 DMA2D foreground PFC control register (DMA2D\_FGPFCCR)

Address offset: 0x001C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALPHA[7:0]								Res.	Res.	RBS	AI	Res.	Res.	AM[1:0]	
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
CS[7:0]								Res.	Res.	START	CCM	CM[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rs	rw	rw	rw	rw	rw

Bits 31:24 **ALPHA[7: 0]**: Alpha value

These bits define a fixed alpha channel value which can replace the original alpha value or be multiplied by the original alpha value according to the alpha mode selected through the AM[1:0] bits.

These bits can only be written when data transfers are disabled. Once a transfer has started, they become read-only.

Bits 23:22 Reserved, must be kept at reset value

Bit 21 **RBS**: Red Blue Swap

This bit allows to swap the R & B to support BGR or ABGR color formats. Once the transfer has started, this bit is read-only.

0: Regular mode (RGB or ARGB)

1: Swap mode (BGR or ABGR)

Bit 20 **AI**: AI: Alpha Inverted

This bit inverts the alpha value. Once the transfer has started, this bit is read-only.

0: Regular alpha

1: Inverted alpha

Bits 19:18 Reserved, must be kept at reset value

Bits 17:16 **AM[1: 0]**: Alpha mode

These bits select the alpha channel value to be used for the foreground image. They can only be written data the transfer are disabled. Once the transfer has started, they become read-only.

00: No modification of the foreground image alpha channel value

01: Replace original foreground image alpha channel value by ALPHA[7: 0]

10: Replace original foreground image alpha channel value by ALPHA[7:0] multiplied with original alpha channel value

other configurations are meaningless

Bits 15:8 **CS[7: 0]**: CLUT size

These bits define the size of the CLUT used for the foreground image. Once the CLUT transfer has started, this field is read-only.

The number of CLUT entries is equal to CS[7:0] + 1.

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

**Bit 5 START:** Start

This bit can be set to start the automatic loading of the CLUT. It is automatically reset:

- at the end of the transfer
- when the transfer is aborted by the user application by setting the ABORT bit in DMA2D\_CR
- when a transfer error occurs
- when the transfer has not started due to a configuration error or another transfer operation already ongoing (data transfer or automatic background CLUT transfer).

**Bit 4 CCM:** CLUT color mode

This bit defines the color format of the CLUT. It can only be written when the transfer is disabled. Once the CLUT transfer has started, this bit is read-only.

0: ARGB8888

1: RGB888

others: meaningless

**Bits 3:0 CM[3: 0]:** Color mode

These bits defines the color format of the foreground image. They can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

0000: ARGB8888

0001: RGB888

0010: RGB565

0011: ARGB1555

0100: ARGB4444

0101: L8

0110: AL44

0111: AL88

1000: L4

1001: A8

1010: A4

others: meaningless

### 13.5.9 DMA2D foreground color register (DMA2D\_FGCOLR)

Address offset: 0x0020

Reset value: 0x0000 0000

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

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

Bits 23:16 **RED[7: 0]**: Red Value

These bits defines the red value for the A4 or A8 mode of the foreground image. They can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Used also for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.

Bits 15:8 **GREEN[7: 0]**: Green Value

These bits defines the green value for the A4 or A8 mode of the foreground image. They can only be written when data transfers are disabled. Once the transfer has started, They are read-only.

Used also for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.

Bits 7:0 **BLUE[7: 0]**: Blue Value

These bits defines the blue value for the A4 or A8 mode of the foreground image. They can only be written when data transfers are disabled. Once the transfer has started, They are read-only.

Used also for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.

**13.5.10 DMA2D background PFC control register (DMA2D\_BGPFCCR)**

Address offset: 0x0024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALPHA[7:0]								Res.	Res.	RBS	AI	Res.	Res.	AM[1:0]	
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
CS[7:0]								Res.	Res.	START	CCM	CM[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rs	rw	rw	rw	rw	rw

Bits 31:24 **ALPHA[7: 0]**: Alpha value

These bits define a fixed alpha channel value which can replace the original alpha value or be multiplied with the original alpha value according to the alpha mode selected with bits AM[1: 0]. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Bits 23:22 Reserved, must be kept at reset value

Bit 21 **RBS**: Red Blue Swap

This bit allows to swap the R & B to support BGR or ABGR color formats. Once the transfer has started, this bit is read-only.

0: Regular mode (RGB or ARGB)

1: Swap mode (BGR or ABGR)

Bit 20 **AI**: AI: Alpha Inverted

This bit inverts the alpha value. Once the transfer has started, this bit is read-only.

0: Regular alpha

1: Inverted alpha

Bits 19:18 Reserved, must be kept at reset value

Bits 17:16 **AM[1: 0]**: Alpha mode

These bits define which alpha channel value to be used for the background image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

00: No modification of the foreground image alpha channel value

01: Replace original background image alpha channel value by ALPHA[7: 0]

10: Replace original background image alpha channel value by ALPHA[7:0] multiplied with original alpha channel value

others: meaningless

Bits 15:8 **CS[7: 0]**: CLUT size

These bits define the size of the CLUT used for the BG. Once the CLUT transfer has started, this field is read-only.

The number of CLUT entries is equal to CS[7:0] + 1.

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



**Bit 5 START:** Start

This bit is set to start the automatic loading of the CLUT. This bit is automatically reset:

- at the end of the transfer
- when the transfer is aborted by the user application by setting the ABORT bit in the DMA2D\_CR
- when a transfer error occurs
- when the transfer has not started due to a configuration error or another transfer operation already on going (data transfer or automatic foreground CLUT transfer).

**Bit 4 CCM:** CLUT Color mode

These bits define the color format of the CLUT. This register can only be written when the transfer is disabled. Once the CLUT transfer has started, this bit is read-only.

0: ARGB8888  
1: RGB888  
others: meaningless

**Bits 3:0 CM[3: 0]:** Color mode

These bits define the color format of the foreground image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

0000: ARGB8888  
0001: RGB888  
0010: RGB565  
0011: ARGB1555  
0100: ARGB4444  
0101: L8  
0110: AL44  
0111: AL88  
1000: L4  
1001: A8  
1010: A4  
others: meaningless

### 13.5.11 DMA2D background color register (DMA2D\_BGCOLR)

Address offset: 0x0028

Reset value: 0x0000 0000

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

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

Bits 23:16 **RED[7: 0]**: Red Value

These bits define the red value for the A4 or A8 mode of the background. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Used also for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.

Bits 15:8 **GREEN[7: 0]**: Green Value

These bits define the green value for the A4 or A8 mode of the background. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Used also for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.

Bits 7:0 **BLUE[7: 0]**: Blue Value

These bits define the blue value for the A4 or A8 mode of the background. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Used also for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.

### 13.5.12 DMA2D foreground CLUT memory address register (DMA2D\_FGCMAR)

Address offset: 0x002C

Reset value: 0x0000 0000

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

Bits 31: 0 **MA[31: 0]**: Memory Address

Address of the data used for the CLUT address dedicated to the foreground image. This register can only be written when no transfer is ongoing. Once the CLUT transfer has started, this register is read-only.

If the foreground CLUT format is 32-bit, the address must be 32-bit aligned.

### 13.5.13 DMA2D background CLUT memory address register (DMA2D\_BGCMAR)

Address offset: 0x0030

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]**: Memory address

Address of the data used for the CLUT address dedicated to the background image.

This register can only be written when no transfer is on going. Once the CLUT transfer has started, this register is read-only.

If the background CLUT format is 32-bit, the address must be 32-bit aligned.

### 13.5.14 DMA2D output PFC control register (DMA2D\_OPFCCR)

Address offset: 0x0034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RBS	AI	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.	Res.	Res.	SB	Res.	Res.	Res.	Res.	Res.	Res.	CM[2:0]		
													rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value

Bit 21 **RBS**: Red Blue Swap

This bit allows to swap the R & B to support BGR or ABGR color formats. Once the transfer has started, this bit is read-only.

- 0: Regular mode (RGB or ARGB)
- 1: Swap mode (BGR or ABGR)

Bit 20 **AI**: Alpha Inverted

This bit inverts the alpha value. Once the transfer has started, this bit is read-only.

- 0: Regular alpha
- 1: Inverted alpha

Bits 19:9 Reserved, must be kept at reset value

Bit 8 **SB**: Swap Bytes

When set, the bytes in the output FIFO are swapped two by two.

When this bit is set, the number of pixel per line (PL) must be even, and the output memory address (OMAR) must be even.

This register can only be written when the transfer is disabled. Once the transfer has started, this register is read-only.

- 0: Bytes in regular order in the output FIFO
- 1: Bytes are swapped two by two in the output FIFO

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

Bits 2:0 **CM[2: 0]**: Color mode

These bits define the color format of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

- 000: ARGB8888
- 001: RGB888
- 010: RGB565
- 011: ARGB1555
- 100: ARGB4444
- others: meaningless

### 13.5.15 DMA2D output color register (DMA2D\_OCOLR)

Address offset: 0x0038

Reset value: 0x0000 0000

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

Bits 31:24 **ALPHA[7: 0]**: Alpha Channel Value

These bits define the alpha channel of the output color. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Bits 23:16 **RED[7: 0]**: Red Value

These bits define the red value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Bits 15:8 **GREEN[7: 0]**: Green Value

These bits define the green value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

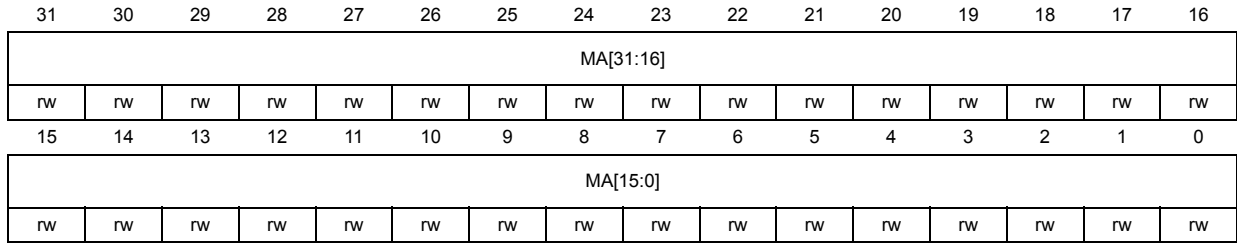
Bits 7:0 **BLUE[7: 0]**: Blue Value

These bits define the blue value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

**13.5.16 DMA2D output memory address register (DMA2D\_OMAR)**

Address offset: 0x003C

Reset value: 0x0000 0000



Bits 31: 0 **MA[31: 0]**: Memory Address

Address of the data used for the output FIFO. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

The address alignment must match the image format selected e.g. a 32-bit per pixel format must be 32-bit aligned and a 16-bit per pixel format must be 16-bit aligned.

### 13.5.17 DMA2D output offset register (DMA2D\_OOR)

Address offset: 0x0040

Reset value: 0x0000 0000

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

The line offset used for the output expressed in pixel when the LOM bit is reset and in byte when the LOM bit is set.

When expressed in pixels, only LO[13:0] is considered, LO[15:14] are ignored.

This value is used for the address generation. It is added at the end of each line to determine the starting address of the next line.

These bits can only be written when data transfers are disabled. Once data transfer has started, they become read-only.

### 13.5.18 DMA2D number of line register (DMA2D\_NLR)

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.	PL[13:0]													
		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
NL[15:0]															
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 29:16 **PL[13: 0]**: Pixel per lines

Number of pixels per lines of the area to be transferred. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

If any of the input image format is 4-bit per pixel, pixel per lines must be even.

Bits 15:0 **NL[15: 0]**: Number of lines

Number of lines of the area to be transferred. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

### 13.5.19 DMA2D line watermark register (DMA2D\_LWR)

Address offset: 0x0048

Reset value: 0x0000 0000

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

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

Bits 15:0 **LW[15:0]**: Line watermark

These bits allow to configure the line watermark for interrupt generation.

An interrupt is raised when the last pixel of the watermarked line has been transferred.

These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

### 13.5.20 DMA2D AHB master timer configuration register (DMA2D\_AMTCR)

Address offset: 0x004C

Reset value: 0x0000 0000

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

Bits 31:16 Reserved

Bits 15:8 **DT[7: 0]**: Dead Time

Dead time value in the AHB clock cycle inserted between two consecutive accesses on the AHB master port. These bits represent the minimum guaranteed number of cycles between two consecutive AHB accesses.

Bits 7:1 Reserved

Bit 0 **EN**: Enable

Enables the dead time functionality.



### 13.5.21 DMA2D register map

The following table summarizes the DMA2D registers. Refer to [Section 2.2.2 on page 82](#) for the DMA2D register base address.

**Table 66. DMA2D 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	
0x0000	DMA2D_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[2:0]	Res.	Res.	Res.	CEIE	CTCIE	CAEIE	TWIE	TCIE	TEIE	Res.	Res.	LOM	Res.	Res.	Res.	Res.	Res.	
	Reset value														0	0	0				0	0	0	0	0	0	0	0						
0x0004	DMA2D_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEIF	CTCIF	CAEIF	TWIF	CTCIF	TEIF
	Reset value																												0	0	0	0	0	0
0x0008	DMA2D_IFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCEIF	CCTCIF	CAECIF	CTWIF	CTCIF	CTEIF	
	Reset value																											0	0	0	0	0	0	
0x000C	DMA2D_FGMAR	MA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0010	DMA2D_FGOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																	
0x0014	DMA2D_BGMAR	MA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0018	DMA2D_BGOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																	
0x001C	DMA2D_FGPFCCR	ALPHA[7:0]							Res.	Res.	RBS	AI	Res.	Res.	AM[1:0]	CS[7:0]							Res.	Res.	START	CCM	CM[3:0]							
	Reset value	0	0	0	0	0	0	0			0	0			0	0																		
0x0020	DMA2D_FGCOLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																	
0x0024	DMA2D_BGPFCCR	ALPHA[7:0]							Res.	Res.	RBS	AI	Res.	Res.	AM[1:0]	CS[7:0]							Res.	Res.	START	CCM	CM[3:0]							
	Reset value	0	0	0	0	0	0	0			0	0			0	0																		
0x0028	DMA2D_BGCOLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																	
0x002C	DMA2D_FGCMAR	MA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0030	DMA2D_BGCMAR	MA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0034	DMA2D_OPFCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RBS	AI	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value											0	0																					



Table 66. DMA2D 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		
0x0038	DMA2D_OCCLR	ALPHA[7:0]							RED[7:0]							GREEN[7:0]					BLUE[7:0]														
		Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RED[4:0]				GREEN[5:0]				BLUE[4:0]									
		Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	A	RED[4:0]			GREEN[4:0]				BLUE[4:0]									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x003C	DMA2D_OMAR	MA[31:0]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0040	DMA2D_OOR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	LO[15:0]																	
	Reset value	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		
0x0044	DMA2D_NLR	Res	Res	PL[13:0]													NL[15:0]																		
	Reset value	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			
0x0048	DMA2D_LWR	Res	Res	LW[15:0]																															
	Reset value	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			
0x004C	DMA2D_AMTCR	Res	Res	DT[7:0]														Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	EN
	Reset value	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	0		
0x0050-0x03FC	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			
0x0400-0x07FC	DMA2D_FGCLUT	ALPHA[7:0][255:0]							RED[7:0][255:0]							GREEN[7:0][255:0]					BLUE[7:0][255:0]														
	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		
0x0800-0x0BFF	DMA2D_BGCLUT	RED[7:0][255:0]							GREEN[7:0][255:0]					BLUE[7:0][255:0]																					
	Reset value	Res	Res	Res	Res	Res	Res	Res	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			



## 14 Chrom-GRC™ (GFXMMU)

### 14.1 Introduction

The graphic MMU is a graphical oriented memory management unit aimed to optimize memory usage according to the display shape.

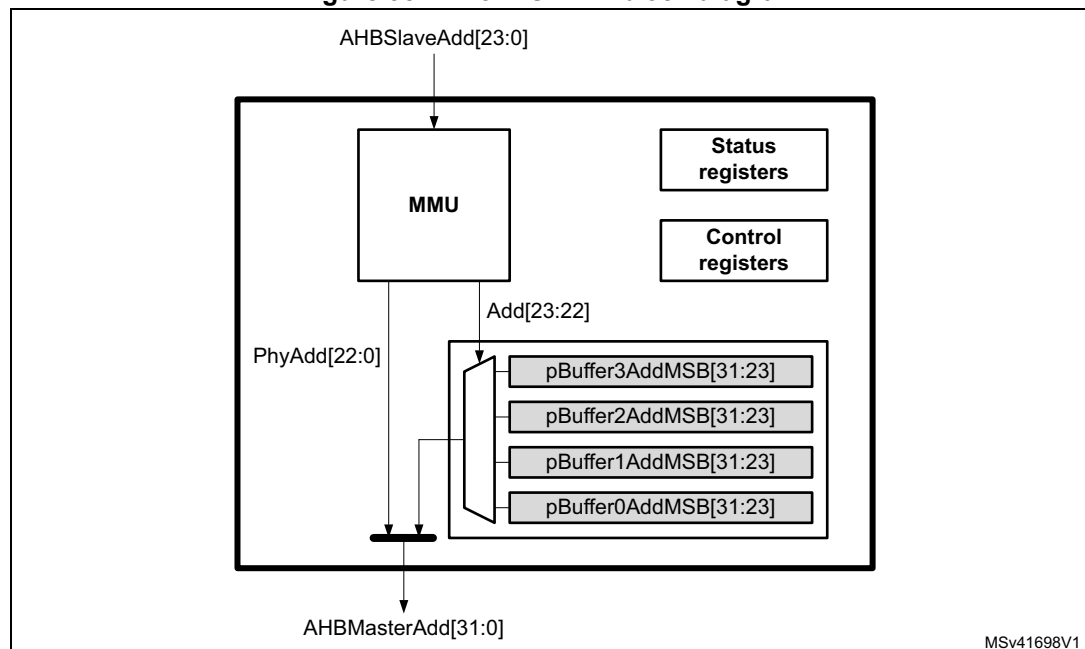
### 14.2 Chrom-GRC™ main features

- Fully programmable display shape to physically store only the visible pixel
- Up to 4 virtual buffer
- Each virtual buffer have 3072 or 4096 bytes per line and 1024 lines
- Each virtual buffer can be physically mapped to any system memory
- Interrupt in case of buffer overflow (1 per buffer)
- Interrupt in case of memory transfer error

## 14.3 Chrom-GRC™ functional and architectural description

The graphic MMU is responsible of address resolution to convert the virtual buffer address into the physical buffer address.

**Figure 35. Chrom-GRC™ block diagram**



### 14.3.1 Virtual memory

The graphic MMU provides a virtual memory space seen by all the system masters. This virtual memory space is divided into four virtual buffers.

#### Virtual buffer

A virtual buffer is seen by any system master as a continuous memory space representing a virtual frame buffer of 1024 lines.

Each line is divided into 192 or 256 16-byte blocks depending on the 192BM bit of the graphic MMU configuration register (GFXMMU\_CR).

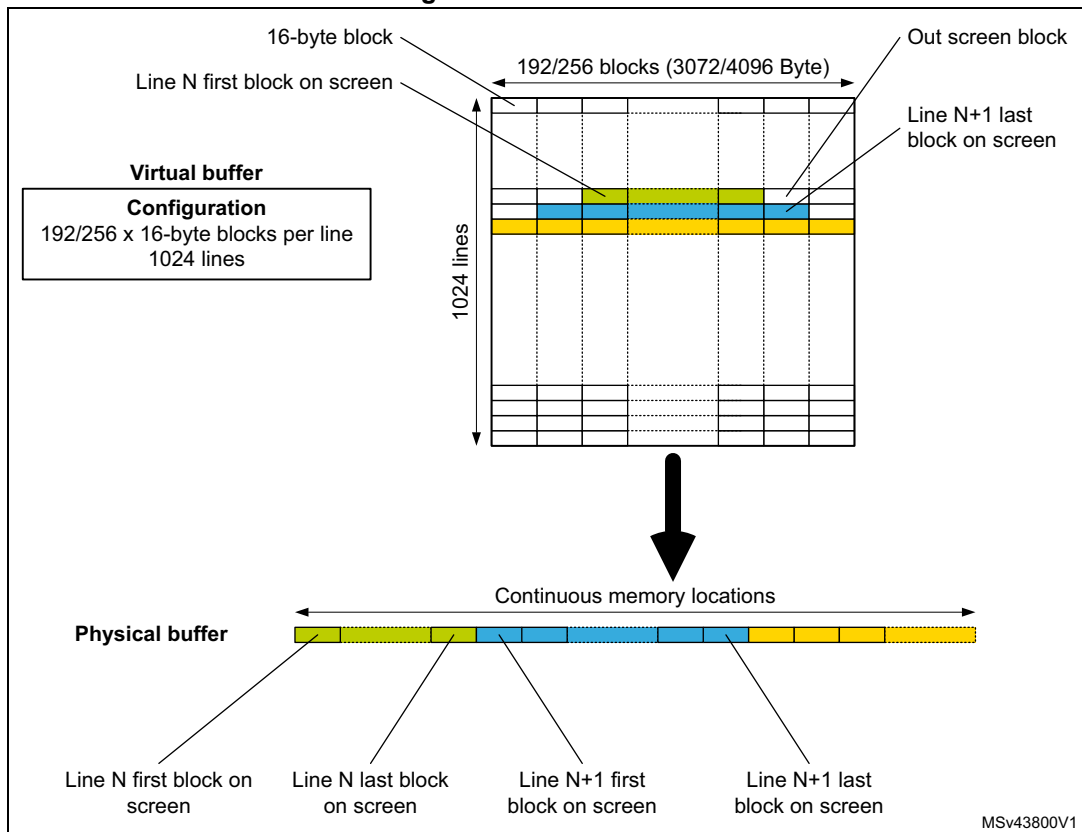
Depending on the display shape and size, only the necessary blocks will be mapped to a physical memory location. This mapping is done programming the LUT entry for each line:

- The enable of the line
- The number of the first “visible” block
- The number of the last “visible” block
- The address offset of the line within the physical buffer

The “visible” blocks can be arranged in the physical buffer in a continuous way programming the address offset of each line.

The LUT is common to all the buffers i.e. all the buffers have the same “shape”.

Figure 36. Virtual buffer



### Virtual buffer overview

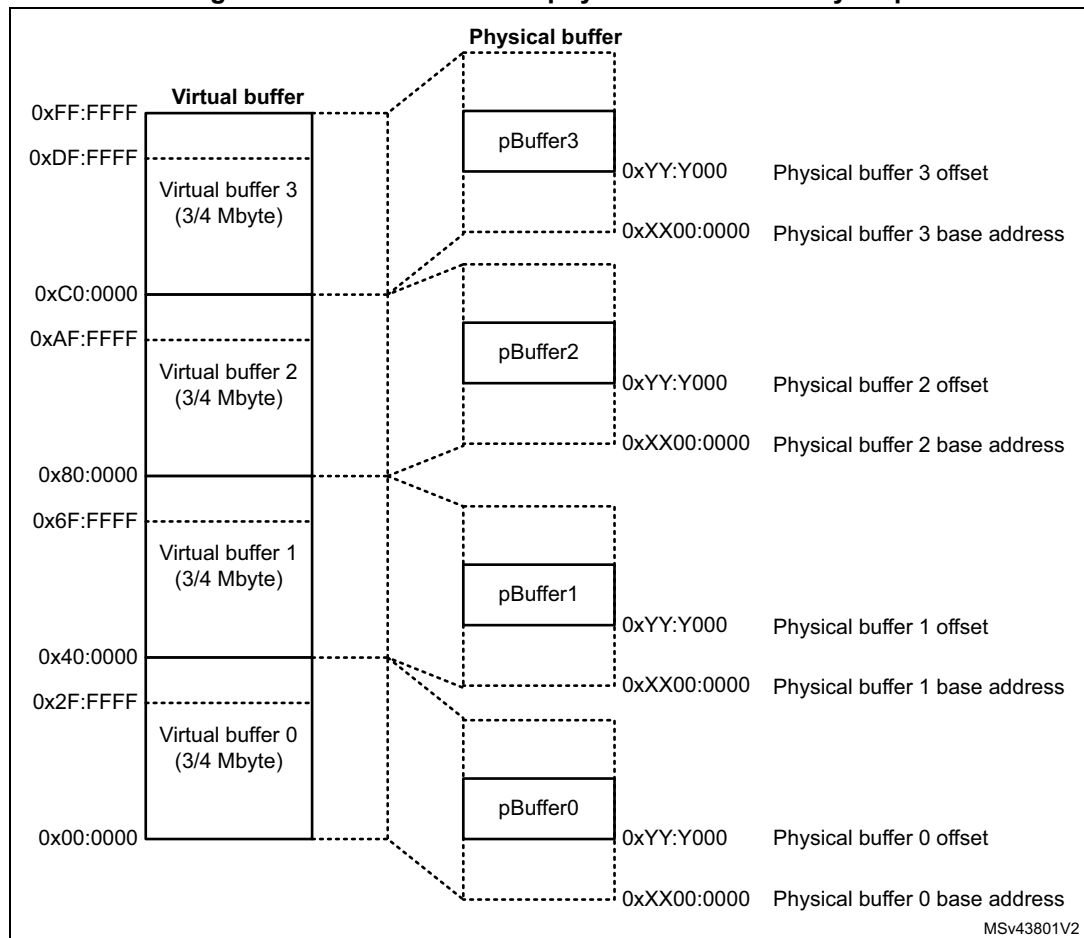
For a frame buffer coded in 32bpp or 16bpp, the virtual buffer can be configured to have 192 or 256 blocks. This will result in a virtual frame buffer of 768 x 1024 or 1024 x 1024 pixels for 32bpp and 1536 x 1024 or 2048 x 1024 for 16bpp.

For a frame buffer coded in 24bpp, the virtual buffer shall be configured to have 192 blocks to have an integer number of pixel per lines. This will result in a virtual frame buffer of 1024 x 1024 pixels for 24bpp.

Each buffer can be physically mapped anywhere in the physical memory thanks to:

- The physical buffer base address (PBBA) field of the graphic MMU buffer x configuration register (GFXMMU\_BxCR). It configures the physical location of the 8 MByte area where the buffer is mapped.
- The physical buffer location relative to the physical buffer base address is defined by the physical buffer offset (PBO) field of the graphic MMU buffer x configuration register (GFXMMU\_BxCR).

Figure 37. Virtual buffer and physical buffer memory map



The buffer can not overflow the 8 MByte boundary of the zone defined by its base address. In case of overflow, the buffer x overflow flag (BxOF) of the graphic MMU status register (GFXMMU\_SR) is set and an interrupt is generated if the buffer x overflow interrupt enable (BxOIE) bit of the graphic MMU configuration register (GFXMMU\_CR) is set.

**Virtual buffer application use case**

As the physical locations are independently configurable, the four virtual buffers can be physically mapped to non continuous locations. This would allow for example to have the four buffers mapped on to four different SDRAM banks and avoid extra precharge cycles accessing the SDRAM.

As a consequence, one buffer shall be used by the CPU/ChromART® for frame buffer calculation while an other one shall be used by the LTDC.

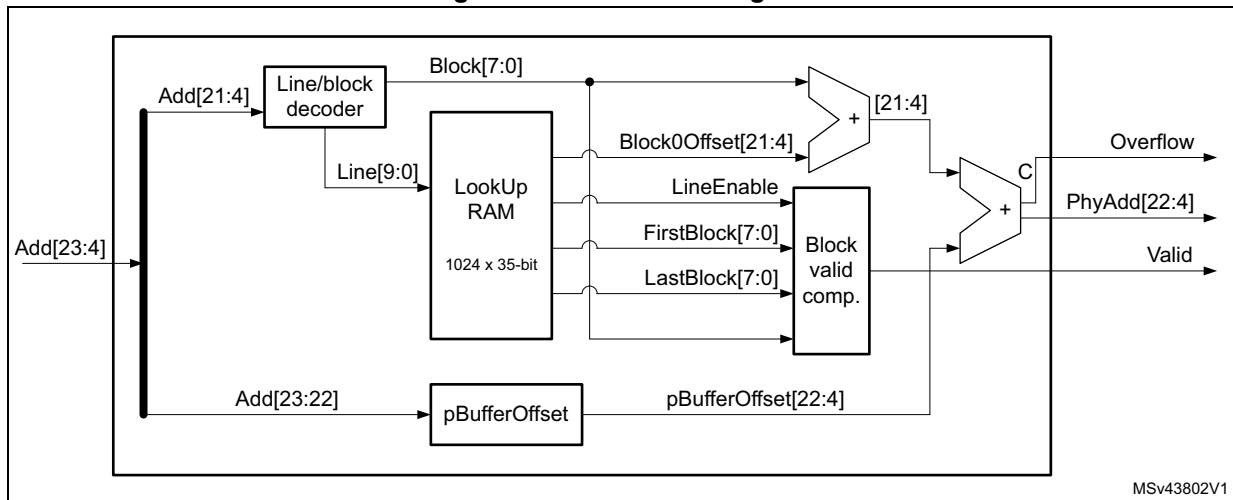
The two remaining buffers can be used as a graphical library requiring extra drawing buffers.

**14.3.2 MMU architecture**

The MMU block is responsible of the address resolution. It receives the 24-bit address and returns the physical 23-bit address and a valid signals to indicate the address is physically mapped or not. The MMU also checks overflow of a area boundary.

The MMU LUT is implemented as a 1024 x 35-bit RAM

**Figure 38. MMU block diagram**



MSv43802V1

### Line block decoder

The line block decoder is generating the block number and the line number according the address.

### Look up RAM

The look up RAM is a 1024 x 35-bit RAM with the following fields:

- 1-bit line enable
- 8-bit first valid block
- 8-bit last valid block
- 18-bit for line offset

As the RAM is bigger than a word, each entry is split into two words on the memory map. The write access are done in two steps:

1. Write the first word with enable/first valid block/last valid block in the GFXMMU\_LUTxL memory location (internally buffered)
2. Write the second word with line offset in the GFXMMU\_LUTxH memory location (effective write into the memory together with the internally buffered value)

A write in the LUT can happen any time but it can lead to inconsistencies if a master is using the MMU at the same time. As the CPU has the priority during LUT programming, this may slow down MMU calculation.

There is no restriction during read operations, but this may slow down CPU as the MMU has the priority on LUT accesses.

### Block validation/comparator

This block is checking is the block is valid.

A block is considered as valid i.e. physically mapped when

- Line is enable
- The block number is greater or equal to the first valid block
- The block number is lower or equal to the last valid block

When the block is valid, the physical address generated is considered as correct.

If the result of the MMU evaluation is not valid, the write operations are ignored, and read operations return the default 32-bit value stored in the default value (DV) field of the graphic MMU default value register (GFXMMU\_DVR).

**Block offset address calculation within the buffer**

The block number is added to the line offset to get the offset of the block within the physical buffer.

As a consequence, the line offset stored in the LUT is given by the following formula:

$$\text{Line offset} = [(\text{Number of visible blocks already used}) - (1^{\text{st}} \text{ visible block})] \times \text{block size}$$

with:

- The maximum value for the line offset is when all the block of all the line are used. As the consequence the line offset for the last line can be maximum:  
 $1023 \times 256 \times 6 = 0x3F:F00x$
- The minimum value for the line offset is when the last block of the first line is the first valid block:  $-255 \times 16 = -0xFFx$  i.e  $0x3F:F01x$

As the consequence the full range of the line offset entry of the LUT is used.

Carry is not taken into account as this stage to be able to perform negative offset calculations (values from  $0x3F:F01x$  to  $0x3F:FFFx$ )

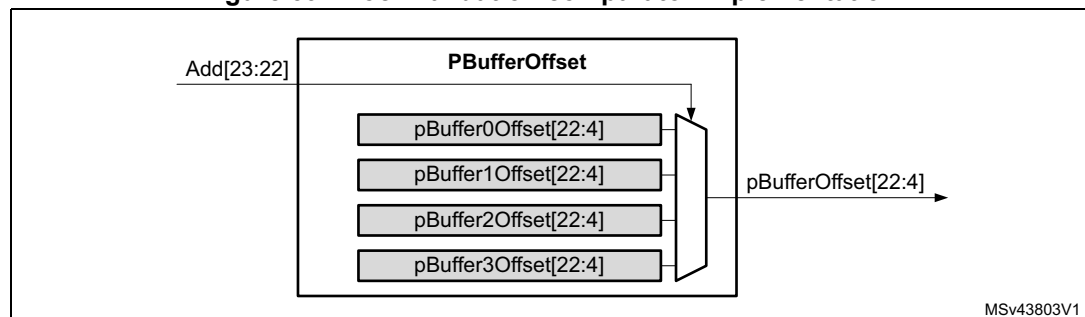
As the block offset is within a 4 MByte buffer, the address generated is 22-bit wide.

**Block offset address calculation**

Once the offset of the block within the buffer as been calculated, this value is added to the offset of the block respective to the physical buffer base address.

The offset of the blocks are defined in registers as shown

**Figure 39. Block validation/comparator implementation**



The resulting address and the buffer offset address shall be on 23-bit.

The carry is taken into account to trigger address overflow. The carry is propagated to the graphic MMU status register (GFXMMU\_SR) to set the buffer x overflow flag (BxOF).



### Example of calculation

We are considering the following configuration for virtual buffer 0

- First visible block of line 0: block 7
- Number of visible block in line 0: 10
- First visible block of line 1: block 6
- Number of visible block in line 1: 12
- Address of the physical buffer: 0xC020:0000

The configuration shall be:

- The base address of the physical buffer 0: 0xC000:0000
- The offset of buffer 0: 0x20:0000
- First visible block of line 0: block 7
- Last visible block of line 0: block 16
- Block 0 offset of line 0:  $(0 - 7) \times 0x10 = -0x70 = 0x3F:FF90$
- First visible block of line 1: block 6
- Last visible block of line 1: block 17
- Block 0 offset of line 1:  $(10 - 6) \times 0x10 = (0xA - 0x6) \times 0x10 = 0x40$

As a consequence:

- the physical address of block 7 of line 0 is:

$$0xC000:0000 + 0x20:0000 + (0x3F:FF90 + 0x70 \text{ without carry}) = 0xC020:0000$$

- the physical address of block 16 of line 0 is:

$$0xC000:0000 + 0x20:0000 + (0x3F:FF90 + 0x100 \text{ without carry}) = 0xC020:0090$$

- the physical address of block 6 of line 1 is:

$$0xC000:0000 + 0x20:0000 + (0x40 + 0x60 \text{ without carry}) = 0xC020:00A0$$

- the physical address of block 17 of line 1 is:

$$0xC000:0000 + 0x20:0000 + (0x40 + 0x110 \text{ without carry}) = 0xC020:0150$$

## 14.4 Graphic MMU interrupts

An interrupt can be produced on the following events:

- Buffer 0 overflow
- Buffer 1 overflow
- Buffer 2 overflow
- Buffer 3 overflow
- AHB master error

Separate interrupt enable bits are available for flexibility.

**Table 67. Graphic MMU interrupt requests**

Interrupt event	Event flag	Enable control bit
Buffer 0 overflow	B0OF	B0OIE
Buffer 1 overflow	B1OF	B1OIE
Buffer 2 overflow	B2OF	B2OIE
Buffer 3 overflow	B3OF	B3OIE
AHB master error	AMEF	AMEIE

## 14.5 Graphic MMU registers

### 14.5.1 Graphic MMU configuration register (GFXMMU\_CR)

Address offset: 0x0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	192BM	Res.	AMEIE	B3OIE	B2OIE	B1OIE	B0OIE
									rw		rw	rw	rw	rw	rw

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

Bit 6 **192BM**: 192 Block mode

This bit defines the number of blocks per line

0: 256 blocks per line

1: 192 blocks per line

Bit 5 Reserved, must be kept at reset value.

Bit 4 **AMEIE**: AHB master error interrupt enable

This bit enables the AHB master error interrupt.

0: Interrupt disable

1: Interrupt enabled

Bit 3 **B3OIE**: Buffer 3 overflow interrupt enable

This bit enables the buffer 3 overflow interrupt.

0: Interrupt disable

1: Interrupt enabled

Bit 2 **B2OIE**: Buffer 2 overflow interrupt enable

This bit enables the buffer 2 overflow interrupt.

0: Interrupt disable

1: Interrupt enabled

Bit 1 **B1OIE**: Buffer 1 overflow interrupt enable

This bit enables the buffer 1 overflow interrupt.

0: Interrupt disable

1: Interrupt enabled

Bit 0 **B0OIE**: Buffer 0 overflow interrupt enable

This bit enables the buffer 0 overflow interrupt.

0: Interrupt disable

1: Interrupt enabled

### 14.5.2 Graphic MMU status register (GFXMMU\_SR)

Address offset: 0x0004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	AMEF	B3OF	B2OF	B1OF	B0OF
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **AMEF**: AHB master error flag

This bit is set when an AHB error happens during a transaction. It is cleared by writing 1 to CAMEF.

Bit 3 **B3OF**: Buffer 3 overflow flag

This bit is set when an overflow occurs during the offset calculation of the buffer 3. It is cleared by writing 1 to CB3OF.

Bit 2 **B2OF**: Buffer 2 overflow flag

This bit is set when an overflow occurs during the offset calculation of the buffer 2. It is cleared by writing 1 to CB2OF.

Bit 1 **B1OF**: Buffer 1 overflow flag

This bit is set when an overflow occurs during the offset calculation of the buffer 1. It is cleared by writing 1 to CB1OF.

Bit 0 **B0OF**: Buffer 0 overflow flag

This bit is set when an overflow occurs during the offset calculation of the buffer 0. It is cleared by writing 1 to CB0OF.

### 14.5.3 Graphic MMU flag clear register (GFXMMU\_FCR)

Address offset: 0x0008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	CAMEF	CB3OF	CB2OF	CB1OF	CB0OF
											rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CAMEF**: Clear AHB master error flag

Writing 1 clears the AHB master error flag in the GFXMMU\_SR register.

Bit 3 **CB3OF**: Clear buffer 3 overflow flag

Writing 1 clears the buffer 3 overflow flag in the GFXMMU\_SR register.

- Bit 2 **CB2OF**: Clear buffer 2 overflow flag  
Writing 1 clears the buffer 2 overflow flag in the GFXMMU\_SR register.
- Bit 1 **CB1OF**: Clear buffer 1 overflow flag  
Writing 1 clears the buffer 1 overflow flag in the GFXMMU\_SR register.
- Bit 0 **CB0OF**: Clear buffer 0 overflow flag  
Writing 1 clears the buffer 0 overflow flag in the GFXMMU\_SR register.

### 14.5.4 Graphic MMU default value register (GFXMMU\_DVR)

Address offset: 0x0010

Reset value: 0x0000 0000

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

Bits 31:0 **DV[31:0]**: Default value  
This field indicates the default 32-bit value which is returned when a master accesses a virtual memory location not physically mapped.

### 14.5.5 Graphic MMU buffer 0 configuration register (GFXMMU\_B0CR)

Address offset: 0x0020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PBBA[31:23]									PBO[22:16]						
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
PBO[15:4]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 **PBBA[31:23]**: Physical buffer base address  
Base address MSB of the physical buffer.

Bits 22:4 **PBO[22:4]**: Physical buffer offset  
Offset of the physical buffer.

Bits 3:0 Reserved, must be kept at reset value.

### 14.5.6 Graphic MMU buffer 1 configuration register (GFXMMU\_B1CR)

Address offset: 0x0024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PBBA[31:23]									PBO[22:16]						
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
PBO[15:4]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:23 **PBBA[31:23]**: Physical buffer base address  
 Base address MSB of the physical buffer.

Bits 22:4 **PBO[22:4]**: Physical buffer offset  
 Offset of the physical buffer.

Bits 3:0 Reserved, must be kept at reset value.

### 14.5.7 Graphic MMU buffer 2 configuration register (GFXMMU\_B2CR)

Address offset: 0x0028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PBBA[31:23]									PBO[22:16]						
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
PBO[15:4]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:23 **PBBA[31:23]**: Physical buffer base address  
 Base address MSB of the physical buffer.

Bits 22:4 **PBO[22:4]**: Physical buffer offset  
 Offset of the physical buffer.

Bits 3:0 Reserved, must be kept at reset value.

### 14.5.8 Graphic MMU buffer 3 configuration register (GFXMMU\_B3CR)

Address offset: 0x002C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PBBA[31:23]									PBO[22:16]						
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
PBO[15:4]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:23 **PBBA[31:23]**: Physical buffer base address  
Base address MSB of the physical buffer.

Bits 22:4 **PBO[22:4]**: Physical buffer offset  
Offset of the physical buffer.

Bits 3:0 Reserved, must be kept at reset value.

### 14.5.9 Graphic MMU LUT entry x low (GFXMMU\_LUTxL)

Address offset: 0x1000 + 8 \* x, x = 0...1023

Reset value: 0x0000 0000

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

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

Bits 23:16 **LVB[7:0]**: Last Valid Block  
Number of the last valid block of line number X.

Bits 15:8 **FVB[7:0]**: First Valid Block  
Number of the first valid block of line number x.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **EN**: Enable  
Line enable.  
0: Line is disabled (no MMU evaluation is performed)  
1: Line is enabled (MMU evaluation is performed)

### 14.5.10 Graphic MMU LUT entry x high (GFXMMU\_LUTxH)

Address offset: 0x1000 + 8 \* x + 4, x = 0...1023

Reset value: 0x0000 0000



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

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:4 **LO[21:4]**: Line offset

Line offset of line number x (i.e. offset of block 0 of line x)

Bits 3:0 Reserved, must be kept at reset value.



### 14.5.11 Graphic MMU register map

The following table summarizes the graphic MMU registers. Refer to the register boundary addresses table for the graphic MMU register base address.

**Table 68. Graphic MMU 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												
0x0000	GFXMMU_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.	19ZBM	Res.	AMEIE	Res.	B3OIE	Res.	B2OIE	Res.	B1OIE	Res.	B0OIE								
	Reset value																											0																	
0x0004	GFXMMU_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AMEF	Res.	B3OF	Res.	B2OF	Res.	B1OF	Res.	B0OF								
	Reset value																																												
0x0008	GFXMMU_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAMEF	Res.	CB3OF	Res.	CB2OF	Res.	CB1OF	Res.	CB0OF								
	Reset value																																												
0x000C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
	Reset value																																												
0x0010	GFXMMU_DRR	DV[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	0	0								
0x0014 0x001C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
	Reset value																																												
0x0020	GFXMMU_B0CR	PBBA[31:23]									PBO[22:4]																		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								
0x0024	GFXMMU_B1CR	PBBA[31:23]									PBO[22:4]																		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								
0x0028	GFXMMU_B2CR	PBBA[31:23]									PBO[22:4]																		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								
0x002C	GFXMMU_B3CR	PBBA[31:23]									PBO[22:4]																		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								
0x0030 0x0FEC 0x0FF0	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.									
	Reset value																																												
0x1000	GFXMMU_LUT0L	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LVB[7:0]					FVB[7:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN								
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x1004	GFXMMU_LUT0H	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LO[21:4]															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								
...																																													
0x2FF8	GFXMMU_LUT1023L	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LVB[7:0]					FVB[7:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN							
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x2FFC	GFXMMU_LUT1023H	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LO[21:4]															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							

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.



## 15 Nested vectored interrupt controller (NVIC)

### 15.1 NVIC main features

- 95 maskable interrupt channels (not including the sixteen Cortex<sup>®</sup>-M4 with FPU interrupt lines)
- 16 programmable priority levels (4 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of System Control Registers

The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming, refer to the PM0214 programming manual for Cortex<sup>™</sup>-M4 products.

### 15.2 SysTick calibration value register

The SysTick calibration value is set to 0x40003A97, which gives a reference time base of 1 ms with the SysTick clock set to 15 MHz ( $\max f_{\text{HCLK}}/8$ ).

## 15.3 Interrupt and exception vectors

The grey rows in [Table 69](#) describe the vectors without specific position.

**Table 69. STM32L4Rxxx and STM32L4Sxxx vector table**

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000 0000
-	-3	fixed	Reset	Reset	0x0000 0004
-	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000 0008
-	-1	fixed	HardFault	All classes of fault	0x0000 000C
-	0	settable	MemManage	Memory management	0x0000 0010
-	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
-	2	settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
-	-	-	-	Reserved	0x0000 001C - 0x0000 0028
-	3	-	-	System service call via SWI instruction	0x0000 002C
-	4	-	-	Monitor	0x0000 0030
-	-	-	-	Reserved	0x0000 0034
-	5	settable	PendSV	Pendable request for system service	0x0000 0038
-	6	settable	SysTick	System tick timer	0x0000 003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000 0040
1	8	settable	PVD_PVM	PVD/PVM1/PVM2/PVM3/PVM4 through EXTI lines 16/35/36/37/38 interrupts	0x0000 0044
2	9	settable	RTC_TAMP_STAMP /CSS_LSE	RTC Tamper or TimeStamp /CSS on LSE through EXTI line 19 interrupts	0x0000 0048
3	10	settable	RTC_WKUP	RTC Wakeup timer through EXTI line 20 interrupt	0x0000 004C
4	11	settable	FLASH	Flash global interrupt	0x0000 0050
5	12	settable	RCC	RCC global interrupt	0x0000 0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000 0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000 005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000 0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000 0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000 0068
11	18	settable	DMA1_CH1	DMA1 channel 1 interrupt	0x0000 006C

Table 69. STM32L4Rxxx and STM32L4Sxxx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
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
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	CAN1_TX	CAN1_TX interrupts	0x0000 008C
20	27	settable	CAN1_RX0	CAN1_RX0 interrupts	0x0000 0090
21	28	settable	CAN1_RX1	CAN1_RX1 interrupt	0x0000 0094
22	29	settable	CAN1_SCE	CAN1_SCE interrupt	0x0000 0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000 009C
24	31	settable	TIM1_BRK/TIM15	TIM1 Break/TIM15 global interrupts	0x0000 00A0
25	32	settable	TIM1_UP/TIM16	TIM1 Update/TIM16 global interrupts	0x0000 00A4
26	33	settable	TIM1_TRG_COM /TIM17	TIM1 trigger and commutation/TIM17 interrupts	0x0000 00A8
27	34	settable	TIM1_CC	TIM1 capture compare interrupt	0x0000 00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000 00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000 00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000 00B8
31	38	settable	I2C1_EV	I2C1 event interrupt	0x0000 00BC
32	39	settable	I2C1_ER	I2C1 error interrupt	0x0000 00C0
33	40	settable	I2C2_EV	I2C2 event interrupt	0x0000 00C4
34	41	settable	I2C2_ER	I2C2 error interrupt	0x0000 00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000 00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000 00D0
37	44	settable	USART1	USART1 global interrupt	0x0000 00D4
38	45	settable	USART2	USART2 global interrupt	0x0000 00D8
39	46	settable	USART3	USART3 global interrupt	0x0000 00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000 00E0
41	48	settable	RTC_ALARM	RTC alarms through EXTI line 18 interrupts	0x0000 00E4
42	49	settable	DFSDM1_FLT3	DFSDM1_FLT3 global interrupt	0x0000 00E8
43	50	settable	TIM8_BRK	TIM8 Break interrupt	0x0000 00EC

Table 69. STM32L4Rxxx and STM32L4Sxxx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
44	51	settable	TIM8_UP	TIM8 Update interrupt	0x0000 00F0
45	52	settable	TIM8_TRG_COM	TIM8 trigger and commutation interrupt	0x0000 00F4
46	53	settable	TIM8_CC	TIM8 capture compare interrupt	0x0000 00F8
47	54	settable	Reserved	Reserved	0x0000 00FC
48	55	settable	FMC	FMC global interrupt	0x0000 0100
49	56	settable	SDMMC1	SDMMC1 global interrupt	0x0000 0104
50	57	settable	TIM5	TIM5 global interrupt	0x0000 0108
51	58	settable	SPI3	SPI3 global interrupt	0x0000 010C
52	59	settable	UART4	UART4 global interrupt	0x0000 0110
53	60	settable	UART5	UART5 global interrupt	0x0000 0114
54	61	settable	TIM6_DACUNDER	TIM6 global and DAC1 underrun interrupts	0x0000 0118
55	62	settable	TIM7	TIM7 global interrupt	0x0000 011C
56	63	settable	DMA2_CH1	DMA2 channel 1 interrupt	0x0000 0120
57	64	settable	DMA2_CH2	DMA2 channel 2 interrupt	0x0000 0124
58	65	settable	DMA2_CH3	DMA2 channel 3 interrupt	0x0000 0128
59	66	settable	DMA2_CH4	DMA2 channel 4 interrupt	0x0000 012C
60	67	settable	DMA2_CH5	DMA2 channel 5 interrupt	0x0000 0130
61	68	settable	DFSDM1_FLT0	DFSDM1_FLT0 global interrupt	0x0000 0134
62	69	settable	DFSDM1_FLT1	DFSDM1_FLT1 global interrupt	0x0000 0138
63	70	settable	DFSDM1_FLT2	DFSDM1_FLT2 global interrupt	0x0000 013C
64	71	settable	COMP	COMP1/COMP2 through EXTI lines 21/22 interrupts	0x0000 0140
65	72	settable	LPTIM1	LPTIM1 global interrupt	0x0000 0144
66	73	settable	LPTIM2	LPTIM2 global interrupt	0x0000 0148
67	74	settable	OTG_FS	OTG_FS global interrupt	0x0000 014C
68	75	settable	DMA2_CH6	DMA2 channel 6 interrupt	0x0000 0150
69	76	settable	DMA2_CH7	DMA2 channel 7 interrupt	0x0000 0154
70	77	settable	LPUART1	LPUART1 global interrupt	0x0000 0158
71	78	settable	OCTOSPI1	OCTOSPI1 global interrupt	0x0000 015C
72	79	settable	I2C3_EV	I2C3 event interrupt	0x0000 0160
73	80	settable	I2C3_ER	I2C3 error interrupt	0x0000 0164
74	81	settable	SAI1	SAI1 global interrupt	0x0000 0168
75	82	settable	SAI2	SAI2 global interrupt	0x0000 016C

Table 69. STM32L4Rxxx and STM32L4Sxxx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
76	83	settable	OCTOSPI2	OCTOSPI2 global interrupt	0x0000 0170
77	84	settable	TSC	TSC global interrupt	0x0000 0174
78	85	settable	DSIHSOT	DSI global interrupt	0x0000 0178
79	86	settable	AES	AES global interrupt	0x0000 017C
80	87	settable	RNG	RNG global interrupt	0x0000 0180
81	88	settable	FPU	Floating point interrupt	0x0000 0184
82	89	settable	HASH and CRS	HASH and CRS interrupt	0x0000 0188
83	90	settable	I2C4_EV	I2C4 event interrupt	0x0000 018C
84	91	settable	I2C4_ER	I2C4 error interrupt	0x0000 0190
85	92	settable	DCMI	DCMI global interrupt	0x0000 0194
86	93	settable	Reserved	Reserved	0x0000 0198
87	94	settable	Reserved	Reserved	0x0000 019C
88	95	settable	Reserved	Reserved	0x0000 01A0
89	96	settable	Reserved	Reserved	0x0000 01A4
90	97	settable	DMA2D	DMA2D global interrupt	0x0000 01A8
91	98	settable	LCD-TFT	LTDC global interrupt	0x0000 019C
92	99	settable	LCD-TFT_ER	LTDC global error interrupt	0x0000 01A0
93	100	settable	GFXMMU	GFXMMU global error interrupt	0x0000 01A4
94	101	settable	DMAMUX1_OVR	DMAMUX Overrun interrupt	0x0000 01A8

## 16 Extended interrupts and events controller (EXTI)

### 16.1 Introduction

The EXTI main features are as follows:

- Generation of up to 39 event/interrupt requests
  - 26 configurable lines
  - 12 direct lines
- Independent mask on each event/interrupt line
- Configurable rising or falling edge (configurable lines only)
- Dedicated status bit (configurable lines only)
- Emulation of event/interrupt requests (configurable lines only)

### 16.2 EXTI main features

The extended interrupts and events controller (EXTI) manages the external and internal asynchronous events/interrupts and generates the event request to the CPU/Interrupt Controller and a wake-up request to the Power Controller.

The EXTI allows the management of up to 39 event lines which can wake up from the Stop 0 and Stop 1 modes. Not all events can wake up from the Stop 2 mode (refer to [Table 70: EXTI lines connections](#)).

The lines are either configurable or direct:

- The lines are configurable: the active edge can be chosen independently, and a status flag indicates the source of the interrupt. The configurable lines are used by the I/Os external interrupts, and by few peripherals.
- The lines are direct: they are used by some peripherals to generate a wakeup from Stop event or interrupt. The status flag is provided by the peripheral.

Each line can be masked independently for an interrupt or an event generation.

This controller also allows to emulate events or interrupts by software, multiplexed with the corresponding hardware event line, by writing to a dedicated register.

### 16.3 EXTI functional description

For the configurable interrupt lines, the interrupt line should be configured and enabled in order to generate an interrupt. This is done by programming the two trigger registers with the desired edge detection and by enabling the interrupt request by writing a '1' to the corresponding bit in the interrupt mask register. When the selected edge occurs on the interrupt line, an interrupt request is generated. The pending bit corresponding to the interrupt line is also set. This request is cleared by writing a '1' in the pending register.

For the direct interrupt lines, the interrupt is enabled by default in the interrupt mask register and there is no corresponding pending bit in the pending register.

To generate an event, the event line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the event request by writing a '1' to the corresponding bit in the event mask register. When the

selected edge occurs on the event line, an event pulse is generated. The pending bit corresponding to the event line is not set.

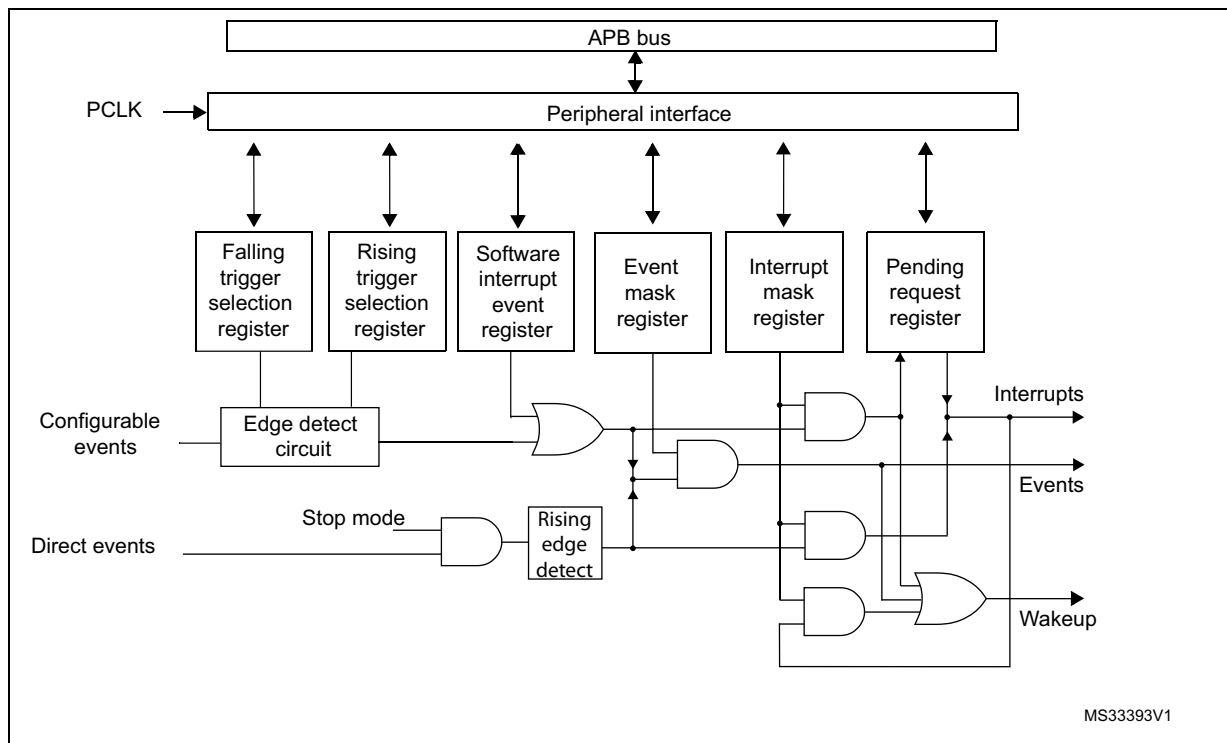
For the configurable lines, an interrupt/event request can also be generated by software by writing a '1' in the software interrupt/event register.

*Note:* The interrupts or events associated to the direct lines are triggered only when the system is in Stop mode. If the system is still running, no interrupt/event is generated by the EXTI.

### 16.3.1 EXTI block diagram

The extended interrupt/event block diagram is shown on [Figure 40](#).

**Figure 40. Configurable interrupt/event block diagram**



### 16.3.2 Wakeup event management

The STM32L4Rxxx and STM32L4Sxxx is able to handle external or internal events in order to wake up the core (WFE). The wakeup event can be generated either by:

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



### 16.3.3 Peripherals asynchronous Interrupts

Some peripherals are able to generate events when the system is in run mode and also when the system is in Stop mode, allowing to wake up the system from Stop mode.

To accomplish this, the peripheral generates both a synchronized (to the system clock, e.g. APB clock) and an asynchronous version of the event. This asynchronous event is connected to an EXTI direct line.

*Note:* Few peripherals with wakeup from Stop capability are connected to an EXTI configurable line. In this case, the EXTI configuration is necessary to allow the wakeup from Stop mode.

### 16.3.4 Hardware interrupt selection

To configure a line as an interrupt source, use the following procedure:

1. Configure the corresponding mask bit in the EXTI\_IMR register.
2. Configure the Trigger Selection bits of the Interrupt line (EXTI\_RTISR and EXTI\_FTISR).
3. Configure the enable and mask bits that control the NVIC IRQ channel mapped to the EXTI so that an interrupt coming from one of the EXTI lines can be correctly acknowledged.

*Note:* The direct lines do not require any EXTI configuration.

### 16.3.5 Hardware event selection

To configure a line as an event source, use the following procedure:

1. Configure the corresponding mask bit in the EXTI\_EMR register.
2. Configure the Trigger Selection bits of the Event line (EXTI\_RTISR and EXTI\_FTISR).

### 16.3.6 Software interrupt/event selection

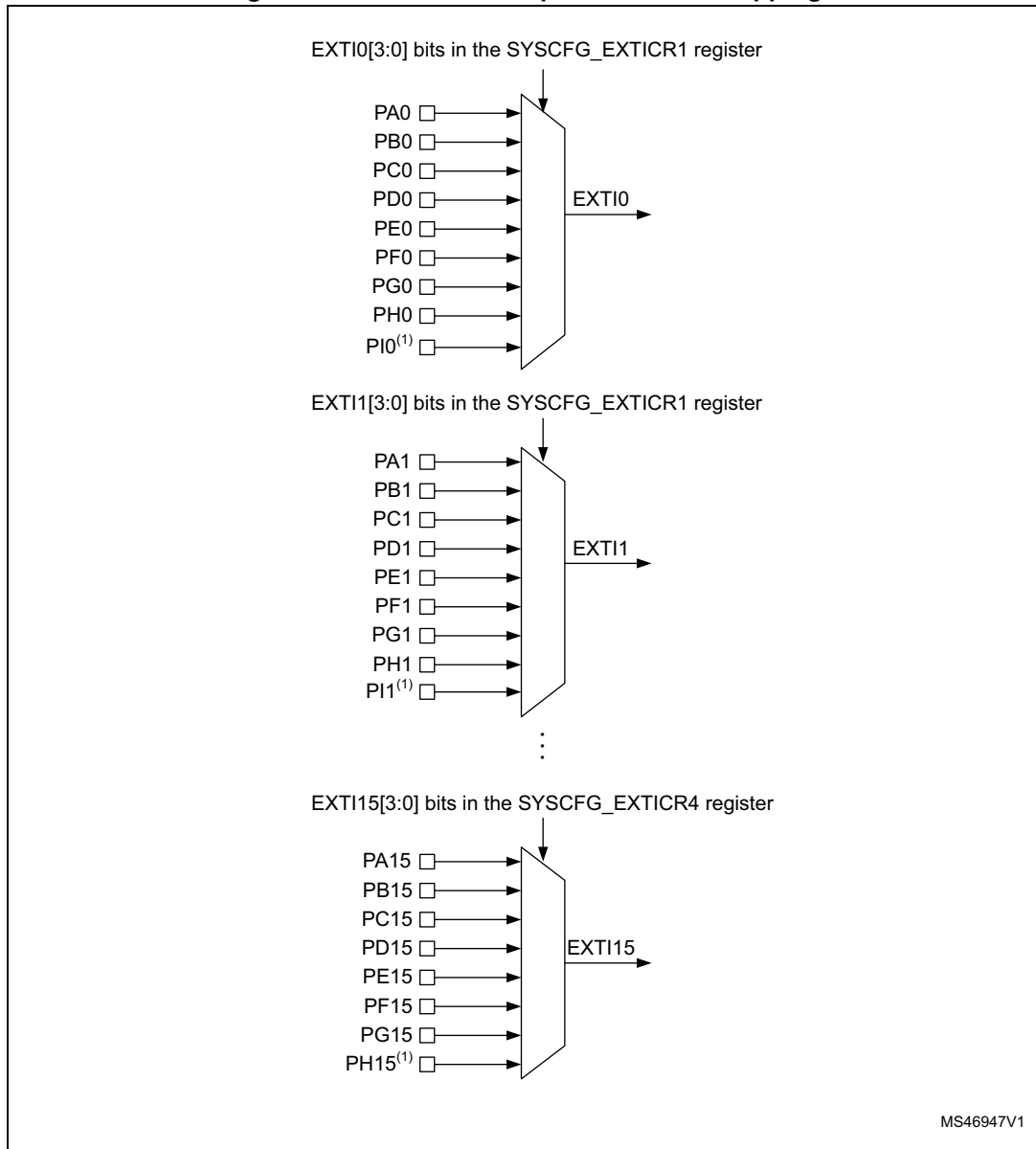
Any of the configurable lines can be configured as a software interrupt/event line. The procedure to generate a software interrupt is as follows:

1. Configure the corresponding mask bit (EXTI\_IMR, EXTI\_EMR).
2. Set the required bit of the software interrupt register (EXTI\_SWIER).

## 16.4 EXTI interrupt/event line mapping

In the STM32L4Rxxx and STM32L4Sxxx, 39 interrupt/event lines are available. The GPIOs are connected to 16 configurable interrupt/event lines (see [Figure 41](#)).

Figure 41. External interrupt/event GPIO mapping



2. The GPIOs PI12 to PI15 are not available.

The EXTI lines are connected as shown in [Table 70: EXTI lines connections](#).

Table 70. EXTI lines connections

EXTI line	Line source <sup>(1)</sup>	Line type
0-15	GPIO	configurable
16	PVD	configurable
17	OTG FS wakeup event <sup>(2)</sup> (OTG_FS_WKUP)	direct
18	RTC alarms	configurable

Table 70. EXTI lines connections (continued)

EXTI line	Line source <sup>(1)</sup>	Line type
19	RTC tamper or timestamp or CSS_LSE	configurable
20	RTC wakeup timer	configurable
21	COMP1 output	configurable
22	COMP2 output	configurable
23	I2C1 wakeup <sup>(2)</sup>	direct
24	I2C2 wakeup <sup>(2)</sup>	direct
25	I2C3 wakeup	direct
26	USART1 wakeup <sup>(2)</sup>	direct
27	USART2 wakeup <sup>(2)</sup>	direct
28	USART3 wakeup <sup>(2)</sup>	direct
29	UART4 wakeup <sup>(2)</sup>	direct
30	UART5 wakeup <sup>(2)</sup>	direct
31	LPUART1 wakeup	direct
32	LPTIM1	direct
33	LPTIM2 <sup>(2)</sup>	direct
34	Reserved	reserved
35	PVM1 wakeup	configurable
36	PVM2 wakeup	configurable
37	PVM3 wakeup	configurable
38	PVM4 wakeup	configurable
39	Reserved	reserved
40	I2C4 wakeup	direct

1. All the lines can wake up from the Stop 0 and Stop 1 modes. All the lines, except the ones mentioned above, can wake up from the Stop 2 mode.
2. This line source cannot wake up from the Stop 2 mode.

## 16.5 EXTI registers

Refer to [Section 1.2 on page 75](#) for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

### 16.5.1 Interrupt mask register 1 (EXTI\_IMR1)

Address offset: 0x00

Reset value: 0xFF82 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **IMx**: Interrupt Mask on line x (x = 31 to 0)

0: Interrupt request from Line x is masked

1: Interrupt request from Line x is not masked

*Note:* The reset value for the direct lines (line 17, lines from 23 to 34, line 39) is set to '1' in order to enable the interrupt by default.

### 16.5.2 Event mask register 1 (EXTI\_EMR1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EM31	EM30	EM29	EM28	EM27	EM26	EM25	EM24	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **EMx**: Event mask on line x (x = 31 to 0)

0: Event request from line x is masked

1: Event request from line x is not masked

### 16.5.3 Rising trigger selection register 1 (EXTI\_RTSR1)

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.	RT22	RT21	RT20	RT19	RT18	Res.	RT16
									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	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 22:18 **RTx**: Rising trigger event configuration bit of line x (x = 22 to 18)

- 0: Rising trigger disabled (for Event and Interrupt) for input line
- 1: Rising trigger enabled (for Event and Interrupt) for input line

Bit 17 Reserved, must be kept at reset value.

Bits 16:0 **RTx**: Rising trigger event configuration bit of line x (x = 16 to 0)

- 0: Rising trigger disabled (for Event and Interrupt) for input line
- 1: Rising trigger enabled (for Event and Interrupt) for input line

*Note:* The configurable wakeup lines are edge-triggered. No glitch must be generated on these lines. If a rising edge on a configurable interrupt line occurs during a write operation in the EXTI\_RTSR register, the pending bit is not set.

Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.

### 16.5.4 Falling trigger selection register 1 (EXTI\_FTSR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT22	FT21	FT20	FT19	FT18	Res.	FT16
									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	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 22:18 **FTx**: Falling trigger event configuration bit of line x (x = 22 to 18)

- 0: Falling trigger disabled (for Event and Interrupt) for input line
- 1: Falling trigger enabled (for Event and Interrupt) for input line

Bit 17 Reserved, must be kept at reset value.

Bits 16:0 **FTx**: Falling trigger event configuration bit of line x (x = 16 to 0)

- 0: Falling trigger disabled (for Event and Interrupt) for input line
- 1: Falling trigger enabled (for Event and Interrupt) for input line

*Note:* The configurable wakeup lines are edge-triggered. No glitch must be generated on these lines. If a falling edge on a configurable interrupt line occurs during a write operation to the EXTI\_FTSR register, the pending bit is not set.  
 Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.

### 16.5.5 Software interrupt event register 1 (EXTI\_SWIER1)

Address offset: 0x10

Reset value: 0x0000 0000

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

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

Bits 22: 18 **SWIx**: Software interrupt on line x (x = 22 o 18)

If the interrupt is enabled on this line in the EXTI\_IMR, writing a '1' to this bit when it is at '0' sets the corresponding pending bit in EXTI\_PR resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit in the EXTI\_PR register (by writing a '1' into the bit).

Bit 17 Reserved, must be kept at reset value.

Bits 16:0 **SWIx**: Software interrupt on line x (x = 16 to 0)

If the interrupt is enabled on this line in the EXTI\_IMR, writing a '1' to this bit when it is at '0' sets the corresponding pending bit in EXTI\_PR resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit of EXTI\_PR (by writing a '1' into the bit).

### 16.5.6 Pending register 1 (EXTI\_PR1)

Address offset: 0x14

Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PIF22	PIF21	PIF20	PIF19	PIF18	Res.	PIF16
									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	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bits 22:18 **PIF<sub>x</sub>**: Pending interrupt flag on line x (x = 22 to 18)

0: No trigger request occurred

1: Selected trigger request occurred

This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.

Bit 17 Reserved, must be kept at reset value.

Bits 16:0 **PIF<sub>x</sub>**: Pending interrupt flag on line x (x = 16 to 0)

0: No trigger request occurred

1: Selected trigger request occurred

This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.

### 16.5.7 Interrupt mask register 2 (EXTI\_IMR2)

Address offset: 0x20

Reset value: 0x0000 0187

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	IM40	IM39	IM38	IM37	IM36	IM35	IM34	IM33	IM32
							rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 8:0 **IM<sub>x</sub>**: Interrupt mask on line x (x = 40 to 32)

0: Interrupt request from line x is masked

1: Interrupt request from line x is not masked

*Note:* The reset value for the direct lines (line 17, lines from 23 to 34, line 39, line 40) is set to '1' in order to enable the interrupt by default.

### 16.5.8 Event mask register 2 (EXTI\_EMR2)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM40	EM39	EM38	EM37	EM36	EM35	EM34	EM33	EM32
							rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 8:0 **EMx**: Event mask on line x (x = 40 to 32)

0: Event request from line x is masked

1: Event request from line x is not masked

### 16.5.9 Rising trigger selection register 2 (EXTI\_RTSR2)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT38	RT37	RT36	RT35	Res.	Res.	Res.
									rw	rw	rw	rw			

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

Bits 7:3 **RTx**: Rising trigger event configuration bit of line x (x = 35 to 38)

0: Rising trigger disabled (for Event and Interrupt) for input line

1: Rising trigger enabled (for Event and Interrupt) for input line

Bits 2:0 Reserved, must be kept at reset value.

*Note: The configurable wakeup lines are edge-triggered. No glitch must be generated on these lines. If a rising edge on a configurable interrupt line occurs during a write operation to the EXTI\_RTISR register, the pending bit is not set.*

*Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.*



### 16.5.10 Falling trigger selection register 2 (EXTI\_FTSR2)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT38	FT37	FT36	FT35	Res.	Res.	Res.
									rw	rw	rw	rw			

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

Bits 7:3 **FTx**: Falling trigger event configuration bit of line x (x = 35 to 38)

0: Falling trigger disabled (for Event and Interrupt) for input line

1: Falling trigger enabled (for Event and Interrupt) for input line

Bits 2:0 Reserved, must be kept at reset value.

*Note:* The configurable wakeup lines are edge-triggered. No glitch must be generated on these lines. If a falling edge on a configurable interrupt line occurs during a write operation to the EXTI\_FTSR register, the pending bit is not set.

Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.

### 16.5.11 Software interrupt event register 2 (EXTI\_SWIER2)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI 38	SWI 37	SWI 36	SWI 35	Res.	Res.	Res.
									rw	rw	rw	rw			

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

Bit 7 **SWIx**: Software interrupt on line x (x = 35 to 38)

If the interrupt is enabled on this line in EXTI\_IMR, writing a '1' to this bit when it is at '0' sets the corresponding pending bit of EXTI\_PR resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit of EXTI\_PR (by writing a '1' to the bit).

Bits 2:0 Reserved, must be kept at reset value.

**16.5.12 Pending register 2 (EXTI\_PR2)**

Address offset: 0x34

Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PIF38	PIF37	PIF36	PIF35	Res.	Res.	Res.
									rc_w1	rc_w1	rc_w1	rc_w1			

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

Bit 7 **PIF<sub>x</sub>**: Pending interrupt flag on line x (x = 35 to 38)

0: No trigger request occurred

1: Selected trigger request occurred

This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' into the bit.

Bits 2:0 Reserved, must be kept at reset value.

16.5.13 EXTI register map

Table 71 gives the EXTI register map and the reset values.

Table 71. Extended interrupt/event controller register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	EXTI_IMR1	IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16	IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0	
	Reset value	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x04	EXTI_EMR1	EM31	EM30	EM29	EM28	EM27	EM26	EM25	EM24	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16	EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x08	EXTI_RTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT22	RT21	RT20	RT19	RT18	Res.	RT16	RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0	
	Reset value										0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0C	EXTI_FTISR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT22	FT21	FT20	FT19	FT18	Res.	FT16	FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0	
	Reset value										0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	EXTI_SWIER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI22	SWI21	SWI20	SWI19	SWI18	Res.	SWI16	SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0	
	Reset value										0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	EXTI_PR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PIF22	PIF21	PIF20	PIF19	PIF18	Res.	PIF16	PIF15	PIF14	PIF13	PIF12	PIF11	PIF10	PIF9	PIF8	PIF7	PIF6	PIF5	PIF4	PIF3	PIF2	PIF1	PIF0	
	Reset value										0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	EXTI_IMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM40	IM39	IM38	IM37	IM36	IM35	IM34	IM33	IM32	
	Reset value																								1	1	0	0	0	0	1	1	1	1
0x24	EXTI_EMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM40	EM39	EM38	EM37	EM36	EM35	EM34	EM33	EM32	
	Reset value																								0	0	0	0	0	0	0	0	0	0
0x28	EXTI_RTISR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT38	RT37	RT36	RT35	Res.	Res.	Res.	
	Reset value																										0	0	0	0				
0x2C	EXTI_FTISR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT38	FT37	FT36	FT35	Res.	Res.	Res.	
	Reset value																										0	0	0	0				
0x30	EXTI_SWIER2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI38	SWI37	SWI36	SWI35	Res.	Res.	Res.	
	Reset value																										0	0	0	0				
0x34	EXTI_PR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PIF38	PIF37	PIF36	PIF35	Res.	Res.	Res.	
	Reset value																										0	0	0	0				

## 17 Cyclic redundancy check calculation unit (CRC)

### 17.1 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.

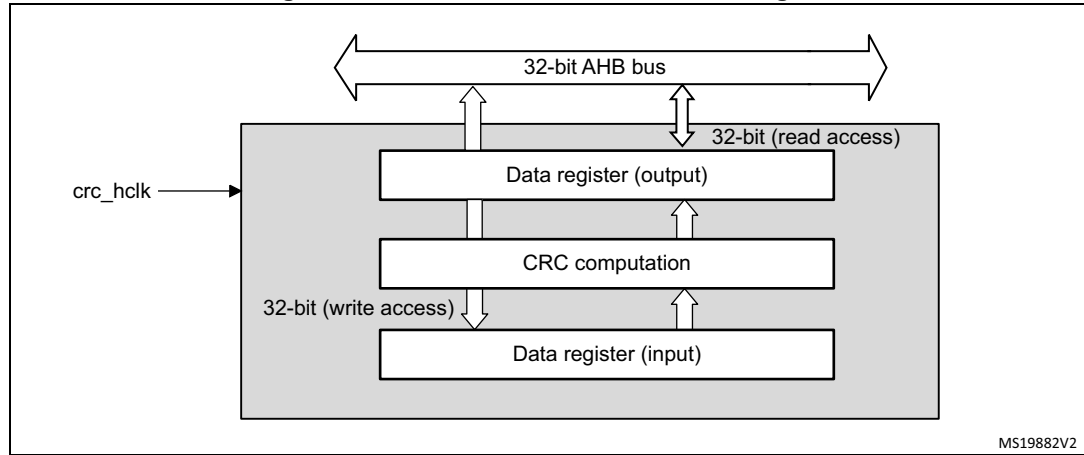
### 17.2 CRC main features

- 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

## 17.3 CRC functional description

### 17.3.1 CRC block diagram

Figure 42. CRC calculation unit block diagram



### 17.3.2 CRC internal signals

Table 72. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

### 17.3.3 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 access is allowed.

The duration of the computation depends on data width:

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

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

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

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

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

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

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

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

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

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

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

### Polynomial programmability

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

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

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

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

## 17.4 CRC registers

### 17.4.1 Data register (CRC\_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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

Bits 31:0 **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.

### 17.4.2 Independent data register (CRC\_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

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

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

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

### 17.4.3 Control register (CRC\_CR)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bits 31:8 Reserved, must be kept 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

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

00: Bit order not affected

01: Bit reversal done by byte

10: Bit reversal done by half-word

11: Bit reversal done by word

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

These bits control the size of the polynomial.

00: 32 bit polynomial

01: 16 bit polynomial

10: 8 bit polynomial

11: 7 bit polynomial

Bits 2:1 Reserved, must be kept 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

### 17.4.4 Initial CRC value (CRC\_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC_INIT[31:16]															
rw	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

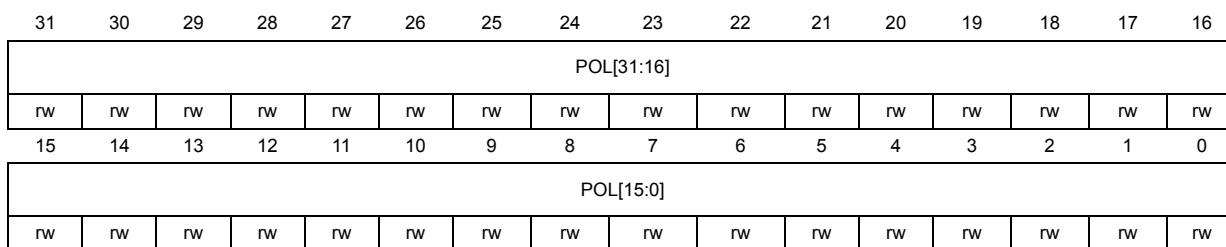


Bits 31:0 **CRC\_INIT[31:0]**: Programmable initial CRC value  
 This register is used to write the CRC initial value.

### 17.4.5 CRC polynomial (CRC\_POL)

Address offset: 0x14

Reset value: 0x04C1 1DB7



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.

### 17.4.6 CRC register map

Table 73. 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	<b>CRC_DR</b>	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	<b>CRC_IDR</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.	IDR[7:0]							
	Reset value																										0	0	0	0	0	0	0
0x08	<b>CRC_CR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]	POLYSIZE[1:0]	Res.	Res.	RESET		
	Reset value																									0	0	0	0	0		0	
0x10	<b>CRC_INIT</b>	CRC_INIT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	<b>CRC_POL</b>	Polynomial coefficients																															
	Reset value	0x04C11DB7																															

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.



## 18 Flexible static memory controller (FSMC)

The Flexible static memory controller (FSMC) includes two memory controllers:

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

### 18.1 FSMC main features

The FSMC functional block makes the interface with: synchronous and asynchronous static memories, and NAND Flash memory. Its main purposes are:

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

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

The main features of the FSMC controller are the following:

- Interface with static-memory mapped devices including:
  - Static random access memory (SRAM)
  - NOR Flash memory/OneNAND Flash memory
  - PSRAM (4 memory banks)
  - Ferroelectric RAM (FRAM)
  - NAND Flash memory with ECC hardware to check up to 8 Kbytes of data
- Interface with parallel LCD modules, supporting Intel 8080 and Motorola 6800 modes.
- Burst mode support for faster access to synchronous devices such as NOR Flash memory, PSRAM)
- Programmable continuous clock output for asynchronous and synchronous accesses
- 8-,16-bit wide data bus
- Independent chip select control for each memory bank
- Independent configuration for each memory bank
- Write enable and byte lane select outputs for use with PSRAM, SRAM devices
- External asynchronous wait control
- Write FIFO with 16 x32-bit depth

The Write FIFO is common to all memory controllers and consists of:

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

The Write FIFO can be disabled by setting the WFDIS bit in the FSMC\_BCR1 register.

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

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

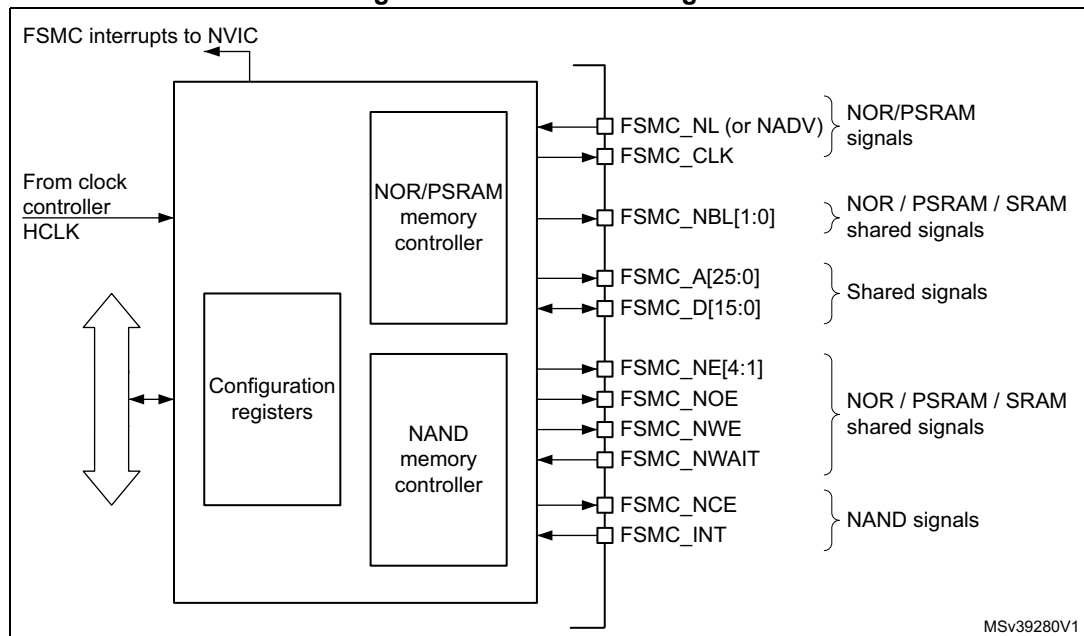
## 18.2 FMC block diagram

The FSMC consists of the following main blocks:

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

The block diagram is shown in the figure below.

Figure 43. FSMC block diagram



## 18.3 AHB interface

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

AHB transactions are translated into the external device protocol. In particular, if the selected external memory is 16- or 8-bit wide, 32-bit wide transactions on the AHB are split into consecutive 16- or 8-bit accesses. The FSMC chip select (FSMC\_NEx) does not toggle between the consecutive accesses except in case of access mode D when the extended mode is enabled.

The FSMC generates an AHB error in the following conditions:

- When reading or writing to an FSMC bank (Bank 1 to 4) which is not enabled.
- When reading or writing to the NOR Flash bank while the FACCEN bit is reset in the FSMC\_BCRx register.

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

- If the access has been attempted by the Cortex<sup>®</sup>-M4 CPU, a hard fault interrupt is generated.
- If the access has been performed by a DMA controller, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

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

### 18.3.1 Supported memories and transactions

#### General transaction rules

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

Therefore, some simple transaction rules must be followed:

- AHB transaction size and memory data size are equal  
There is no issue in this case.
- AHB transaction size is greater than the memory size:  
In this case, the FSMC splits the AHB transaction into smaller consecutive memory accesses to meet the external data width. The FSMC chip select (FSMC\_NEx) does not toggle between the consecutive accesses. If the bus turnaround timings is configured to any other value than 0, the FMC chip select (FMC\_NEx) toggles between

the consecutive accesses. This feature is required when interfacing with FRAM memory.

- AHB transaction size is smaller than the memory size:

The transfer may or not be consistent depending on the type of external device:

- Accesses to devices that have the byte select feature (SRAM, ROM, PSRAM)

In this case, the FSMC allows read/write transactions and accesses the right data through its byte lanes NBL[1:0].

Bytes to be written are addressed by NBL[1:0].

All memory bytes are read (NBL[1:0] are driven low during read transaction) and the useless ones are discarded.

- Accesses to devices that do not have the byte select feature (NOR and NAND Flash memories)

This situation occurs when a byte access is requested to a 16-bit wide Flash memory. Since the device cannot be accessed in byte mode (only 16-bit words can be read/written from/to the Flash memory), Write transactions and Read transactions are allowed (the controller reads the entire 16-bit memory word and uses only the required byte).

### Wrap support for NOR Flash/PSRAM

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

### Configuration registers

The FSMC can be configured through a set of registers. Refer to [Section 18.5.6](#), for a detailed description of the NOR Flash/PSRAM controller registers. Refer to [Section 18.6.7](#), for a detailed description of the NAND Flash registers.

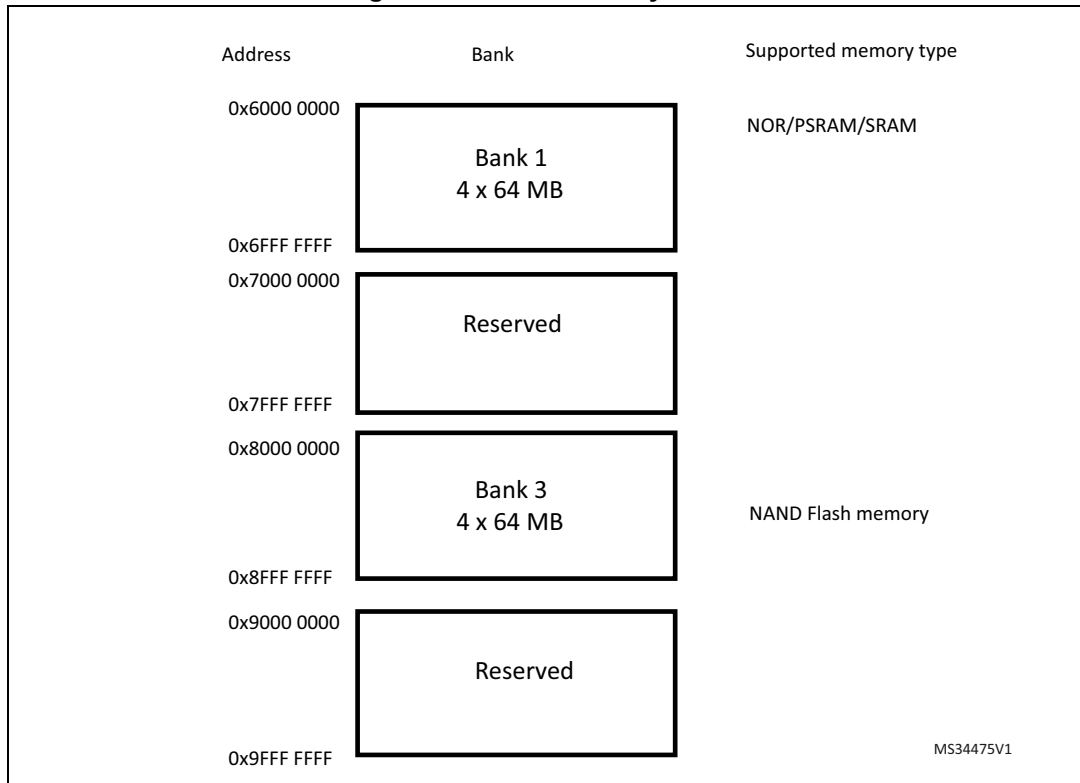
## 18.4 External device address mapping

From the FSMC point of view, the external memory is divided into fixed-size banks of 256 Mbytes each (see [Figure 44](#)):

- Bank 1 used to address up to 4 NOR Flash memory or PSRAM devices. This bank is split into 4 NOR/PSRAM subbanks with 4 dedicated chip selects, as follows:
  - Bank 1 - NOR/PSRAM 1
  - Bank 1 - NOR/PSRAM 2
  - Bank 1 - NOR/PSRAM 3
  - Bank 1 - NOR/PSRAM 4
- Bank 3 used to address NAND Flash memory devices. The MPU memory attribute for this space must be reconfigured by software to Device.

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

Figure 44. FSMC memory banks



### 18.4.1 NOR/PSRAM address mapping

HADDR[27:26] bits are used to select one of the four memory banks as shown in [Table 74](#).

Table 74. NOR/PSRAM bank selection

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

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

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

Table 75. NOR/PSRAM External memory address

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

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

### 18.4.2 NAND Flash memory address mapping

The NAND bank is divided into memory areas as indicated in [Table 76](#).

**Table 76. NAND memory mapping and timing registers**

Start address	End address	FSMC bank	Memory space	Timing register
0x8800 0000	0x8BFF FFFF	Bank 3 - NAND Flash	Attribute	FSMC_PATT (0x8C)
0x8000 0000	0x83FF FFFF		Common	FSMC_PMEM (0x88)

For NAND Flash memory, the common and attribute memory spaces are subdivided into three sections (see in [Table 77](#) below) located in the lower 256 Kbytes:

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

**Table 77. NAND bank selection**

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

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

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

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

## 18.5 NOR Flash/PSRAM controller

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

- Asynchronous SRAM, FRAM and ROM
  - 8 bits
  - 16 bits
- PSRAM (CellularRAM™)
  - Asynchronous mode
  - Burst mode for synchronous accesses
  - Multiplexed or non-multiplexed
- NOR Flash memory
  - Asynchronous mode
  - Burst mode for synchronous accesses
  - Multiplexed or non-multiplexed

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

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

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

The FSMC Clock (FSMC\_CLK) is a submultiple of the HCLK clock. It can be delivered to the selected external device either during synchronous accesses only or during asynchronous and synchronous accesses depending on the CCKEN bit configuration in the FSMC\_BCR1 register:

- If the CCLKEN bit is reset, the FSMC generates the clock (CLK) only during synchronous accesses (Read/write transactions).
- If the CCLKEN bit is set, the FSMC generates a continuous clock during asynchronous and synchronous accesses. To generate the FSMC\_CLK continuous clock, Bank 1 must be configured in synchronous mode (see [Section 18.5.6: NOR/PSRAM controller registers](#)). Since the same clock is used for all synchronous memories, when a continuous output clock is generated and synchronous accesses are performed, the AHB data size has to be the same as the memory data width (MWID) otherwise the FSMC\_CLK frequency will be changed depending on AHB data transaction (refer to [Section 18.5.5: Synchronous transactions](#) for FSMC\_CLK divider ratio formula).

The size of each bank is fixed and equal to 64 Mbytes. Each bank is configured through dedicated registers (see [Section 18.5.6: NOR/PSRAM controller registers](#)).

The programmable memory parameters include access times (see [Table 78](#)) and support for wait management (for PSRAM and NOR Flash accessed in burst mode).



**Table 78. Programmable NOR/PSRAM access parameters**

Parameter	Function	Access mode	Unit	Min.	Max.
Address setup	Duration of the address setup phase	Asynchronous	AHB clock cycle (HCLK)	0	15
Address hold	Duration of the address hold phase	Asynchronous, muxed I/Os	AHB clock cycle (HCLK)	1	15
NBL setup	Duration of the byte lanes setup phase	Asynchronous	AHB clock cycle (HCLK)	0	3
Data setup	Duration of the data setup phase	Asynchronous	AHB clock cycle (HCLK)	1	256
Data hold	Duration of the data hold phase	Asynchronous	AHB clock cycle (HCLK)	0	3
Bust turn	Duration of the bus turnaround phase	Asynchronous and synchronous read / write	AHB clock cycle (HCLK)	0	15
Clock divide ratio	Number of AHB clock cycles (HCLK) to build one memory clock cycle (CLK)	Synchronous	AHB clock cycle (HCLK)	2	16
Data latency	Number of clock cycles to issue to the memory before the first data of the burst	Synchronous	Memory clock cycle (CLK)	2	17

### 18.5.1 External memory interface signals

[Table 79](#), [Table 80](#) and [Table 81](#) list the signals that are typically used to interface with NOR Flash memory, SRAM and PSRAM.

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

#### NOR Flash memory, non-multiplexed I/Os

**Table 79. Non-multiplexed I/O NOR Flash memory**

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

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

**NOR Flash memory, 16-bit multiplexed I/Os****Table 80. 16-bit multiplexed I/O NOR Flash memory**

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

The maximum capacity is 512 Mbits.

**PSRAM/FRAM/SRAM, non-multiplexed I/Os****Table 81. Non-multiplexed I/Os PSRAM/SRAM**

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

The maximum capacity is 512 Mbits.

**PSRAM, 16-bit multiplexed I/Os****Table 82. 16-Bit multiplexed I/O PSRAM**

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

Table 82. 16-Bit multiplexed I/O PSRAM (continued)

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

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

## 18.5.2 Supported memories and transactions

Table 83 below shows an example of the supported devices, access modes and transactions when the memory data bus is 16-bit wide for NOR Flash memory, PSRAM and SRAM. The transactions not allowed (or not supported) by the FSMC are shown in gray in this example.

Table 83. NOR Flash/PSRAM: example of supported memories and transactions

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

Table 83. NOR Flash/PSRAM: example of supported memories and transactions

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

### 18.5.3 General timing rules

#### Signals synchronization

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

### 18.5.4 NOR Flash/PSRAM controller asynchronous transactions

#### Asynchronous static memories (NOR Flash, PSRAM, SRAM, FRAM)

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

#### Mode 1 - SRAM/FRAM/PSRAM (CRAM)

The next figures show the read and write transactions for the supported modes followed by the required configuration of FSMC\_BCRx, and FSMC\_BTRx/FSMC\_BWTRx registers.

Figure 45. Mode1 read access waveforms

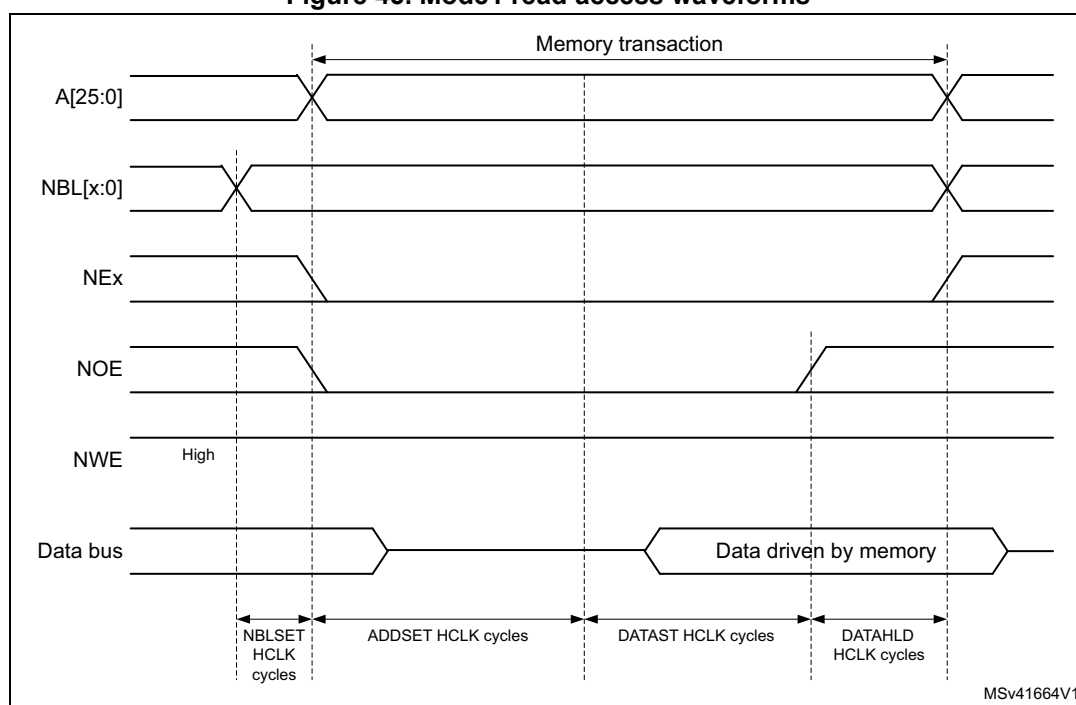
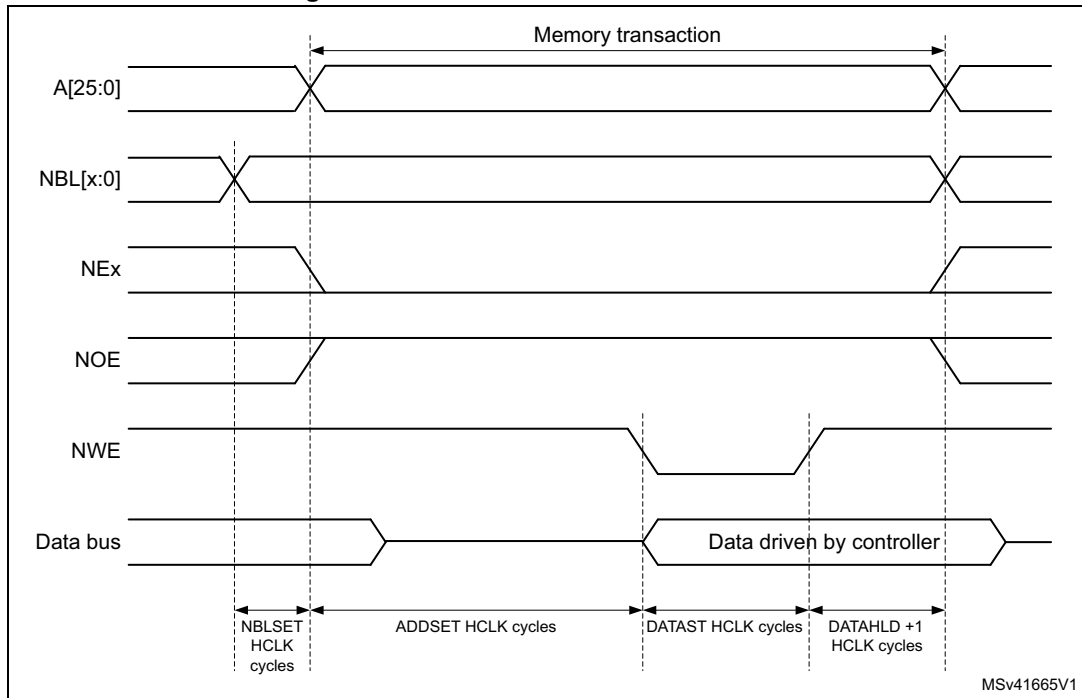


Figure 46. Mode1 write access waveforms



The DATAHLD time at the end of the read and write transactions guarantee the address and data hold time after the NOE/NWE rising edge. The DATAST value must be greater than zero (DATAST > 0).

Table 84. FSMC\_BCRx bit fields

Bit number	Bit name	Value to set
31:24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	Reserved	0x0
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1

Table 84. FSMC\_BCRx bit fields (continued)

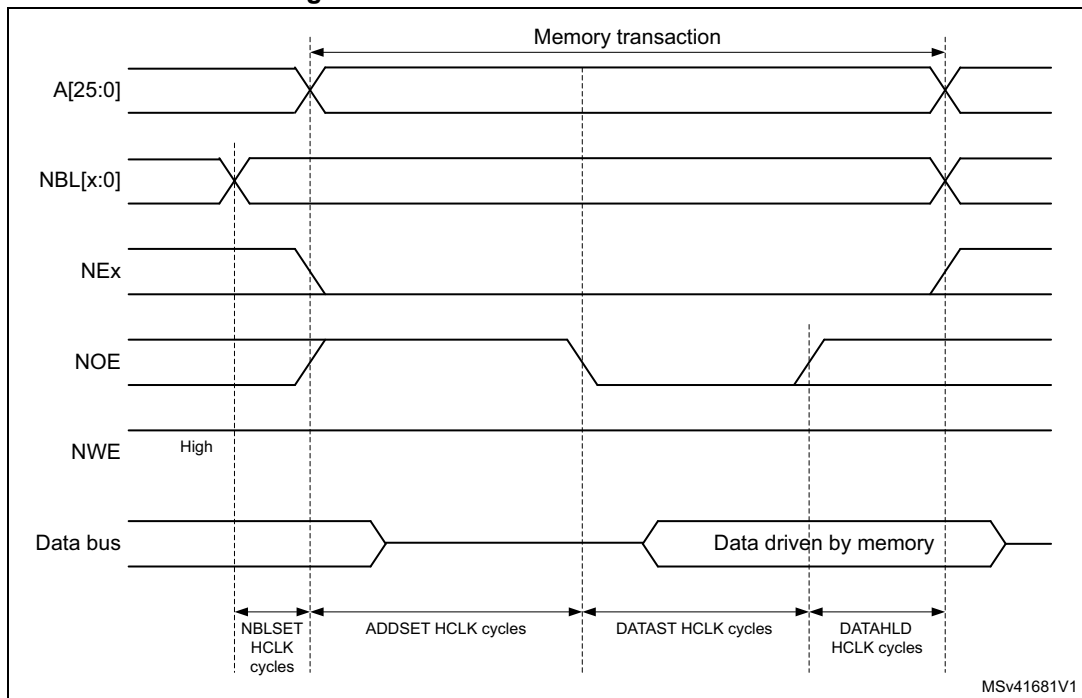
Bit number	Bit name	Value to set
6	FACCEN	Don't care
5:4	MWID	As needed
3:2	MTYP	As needed, exclude 0x2 (NOR Flash memory)
1	MUXE	0x0
0	MBKEN	0x1

Table 85. FSMC\_BTRx bit fields

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

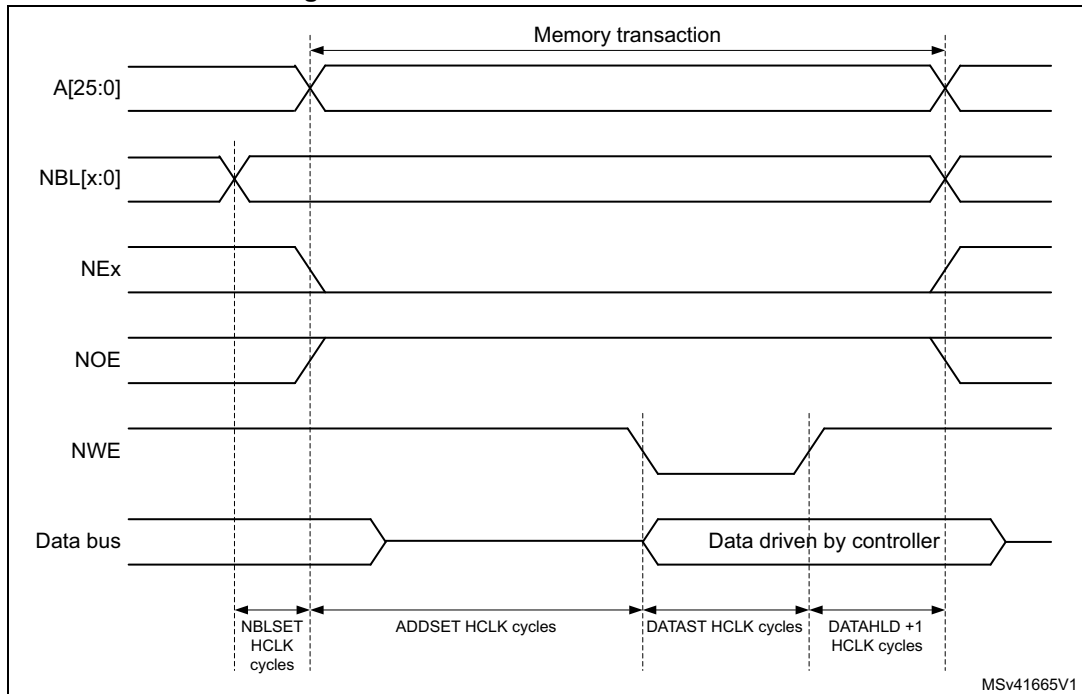
Mode A - SRAM/FRAM/PSRAM (CRAM) OE toggling

Figure 47. ModeA read access waveforms



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

Figure 48. ModeA write access waveforms



The differences compared with Mode1 are the toggling of NOE and the independent read and write timings.



Table 86. FSMC\_BCRx bit fields

Bit number	Bit name	Value to set
31:24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCAWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don't care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Don't care
5:4	MWID	As needed
3:2	MTYP	As needed, exclude 0x2 (NOR Flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 87. FSMC\_BTRx bit fields

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

Table 88. FSMC\_BWTRx bit fields

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

Mode 2/B - NOR Flash

Figure 49. Mode2 and mode B read access waveforms

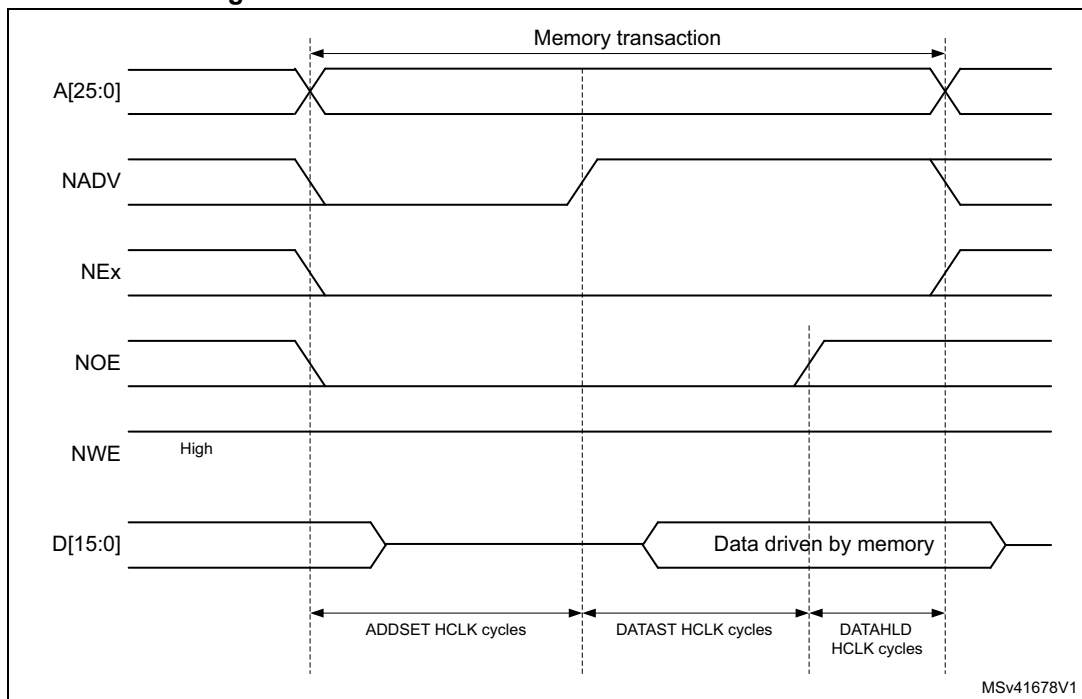


Figure 50. Mode2 write access waveforms

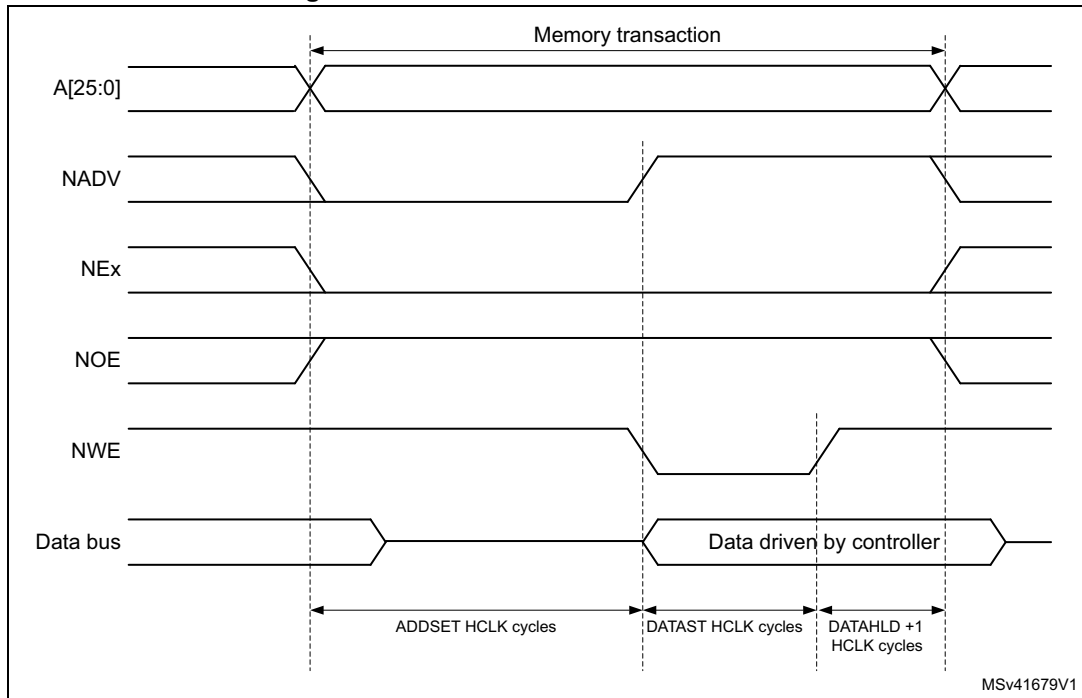
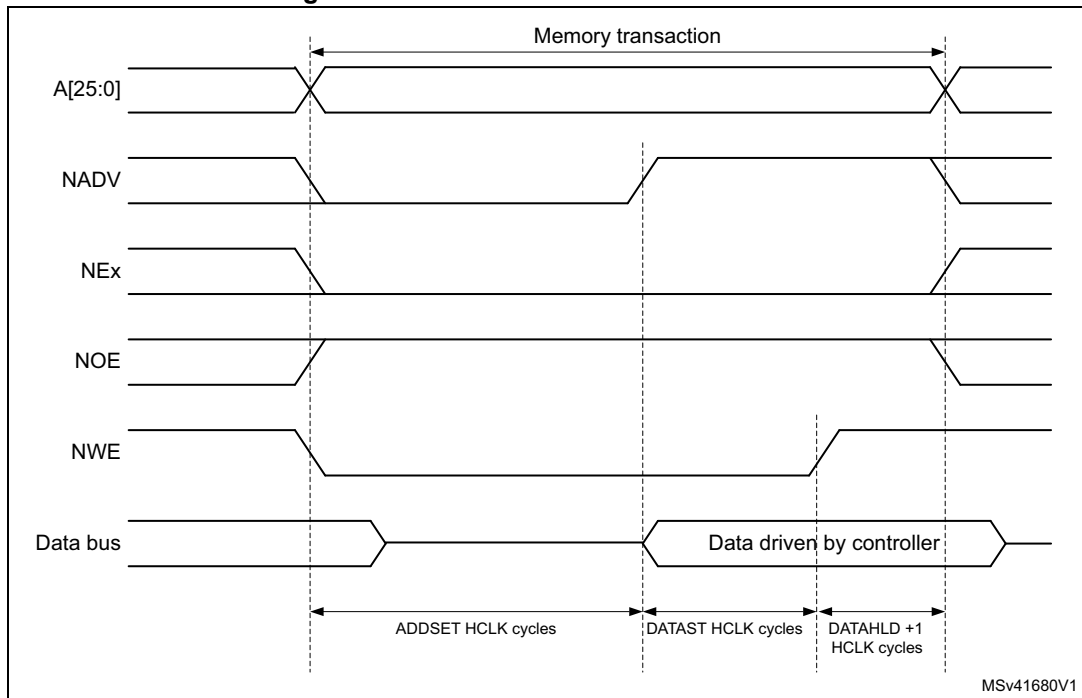


Figure 51. ModeB write access waveforms



The differences with Mode1 are the toggling of NWE and the independent read and write timings when extended mode is set (Mode B).

Table 89. FSMC\_BCRx bit fields

Bit number	Bit name	Value to set
31:24	Reserved	0x000
23:22	NBLSET[1:0]	Don't care
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1 for mode B, 0x0 for mode 2
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don't care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5:4	MWID	As needed
3:2	MTYP	0x2 (NOR Flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 90. FSMC\_BTRx bit fields

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD HCLK cycles for read accesses and DATAHLD+1 HCLK cycles for write accesses when Extended mode is disabled).
29:28	ACCMOD	0x1 if extended mode is set
27:24	DATLAT	Don't care
23:20	CLKDIV	Don't care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the access second phase (DATAST HCLK cycles) for read accesses.
7:4	ADDHLD	Don't care
3:0	ADDSET	Duration of the access first phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 0.

**Table 91. FSMC\_BWTRx bit fields**

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD+1 HCLK cycles for write accesses).
29:28	ACCMOD	0x1 if extended mode is set
27:24	DATLAT	Don't care
23:20	CLKDIV	Don't care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the access second phase (DATAST HCLK cycles) for write accesses.
7:4	ADDHLD	Don't care
3:0	ADDSET	Duration of the access first phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 0.

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

**Mode C - NOR Flash - OE toggling**

**Figure 52. ModeC read access waveforms**

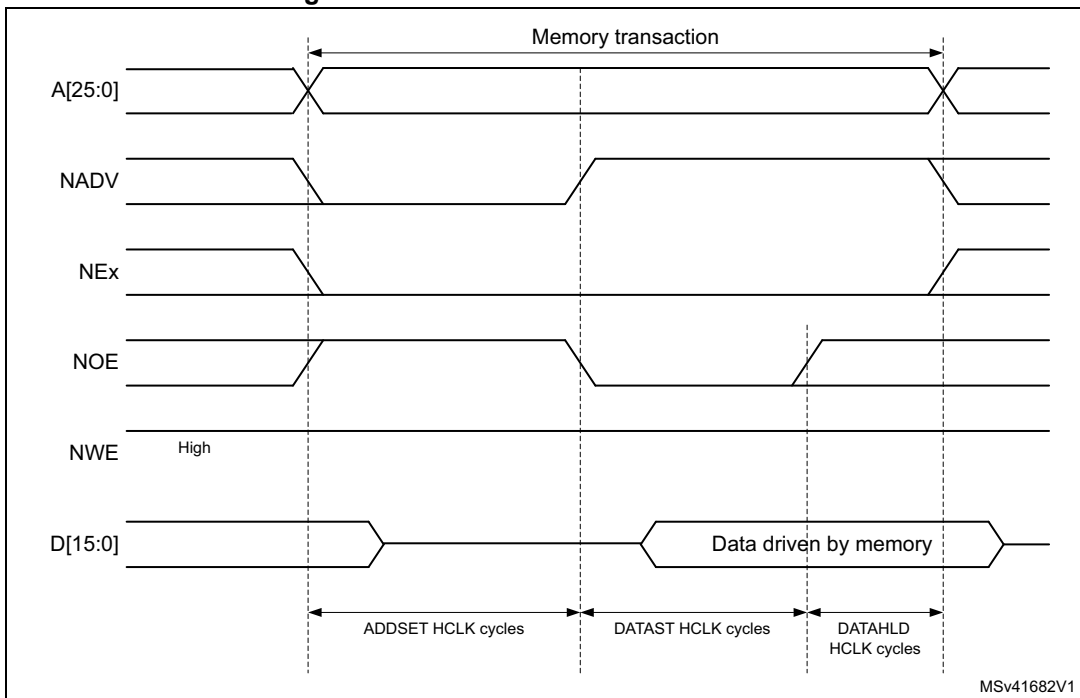
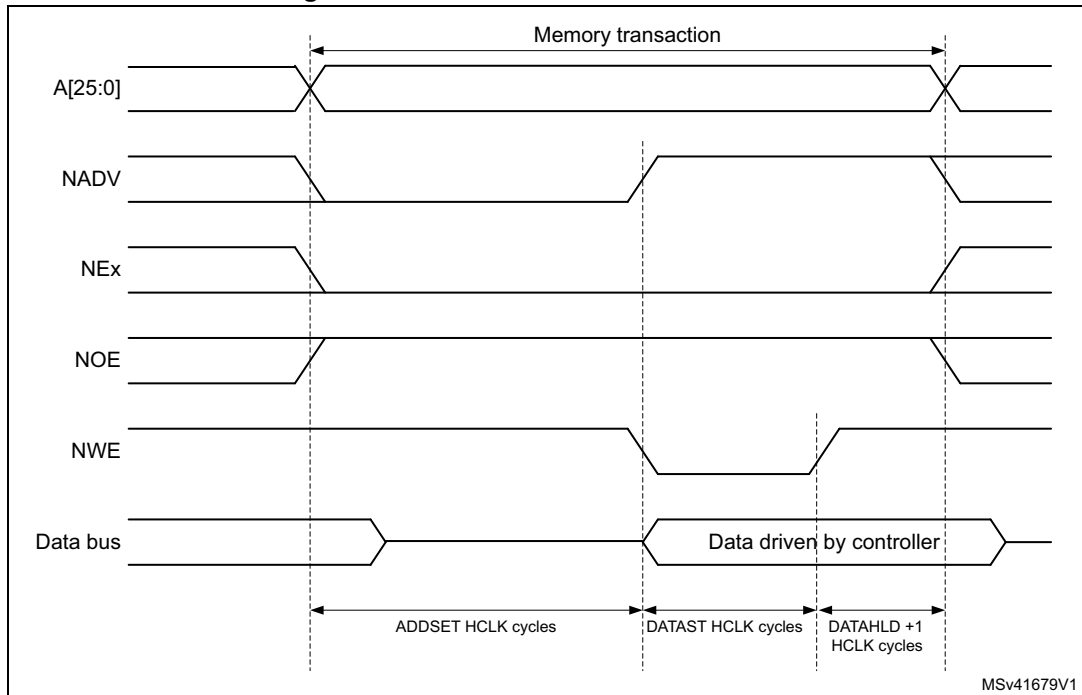


Figure 53. ModeC write access waveforms



The differences compared with Mode1 are the toggling of NOE and the independent read and write timings.

Table 92. FSMC\_BCRx bit fields

Bit number	Bit name	Value to set
31:24	Reserved	0x000
23:22	NBLSET[1:0]	Don't care
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don't care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1

Table 92. FSMC\_BCRx bit fields (continued)

Bit number	Bit name	Value to set
5:4	MWID	As needed
3:2	MTYP	0x02 (NOR Flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 93. FSMC\_BTRx bit fields

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD HCLK cycles for read accesses).
29:28	ACCMOD	0x2
27:24	DATLAT	0x0
23:20	CLKDIV	0x0
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for read accesses.
7:4	ADDHLD	Don't care
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 0.

Table 94. FSMC\_BWTRx bit fields

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD+1 HCLK cycles for write accesses).
29:28	ACCMOD	0x2
27:24	DATLAT	Don't care
23:20	CLKDIV	Don't care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for write accesses.
7:4	ADDHLD	Don't care
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 0.

Mode D - asynchronous access with extended address

Figure 54. ModeD read access waveforms

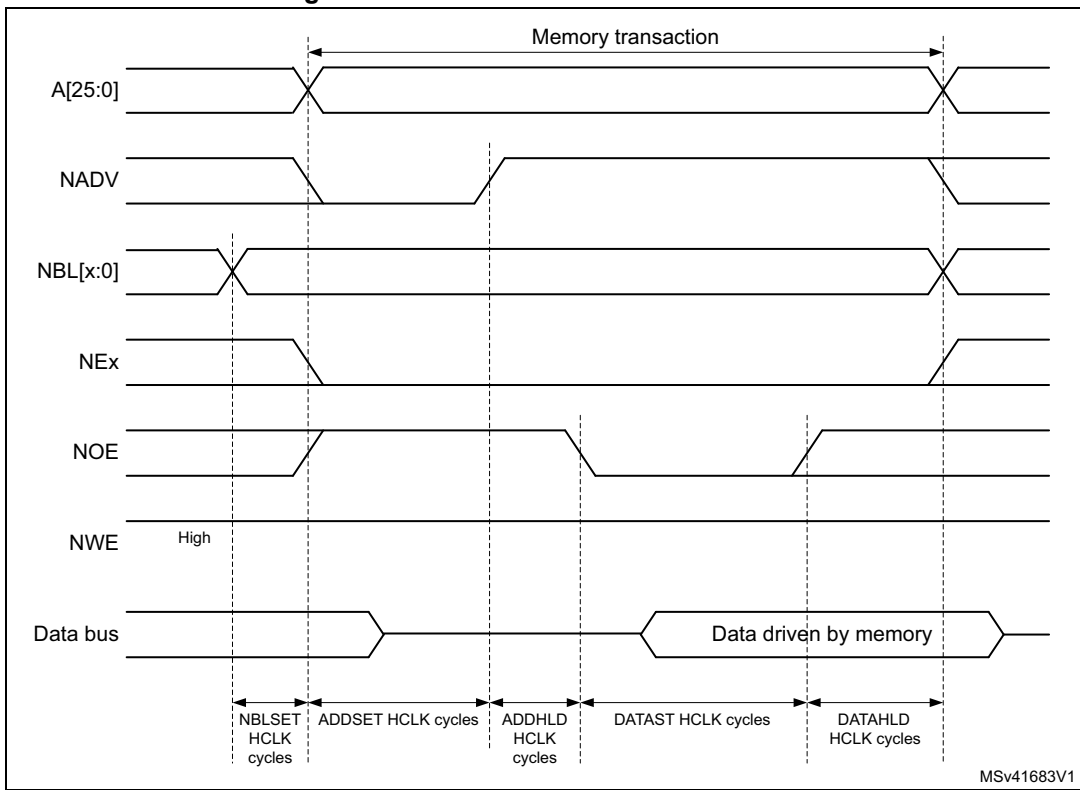
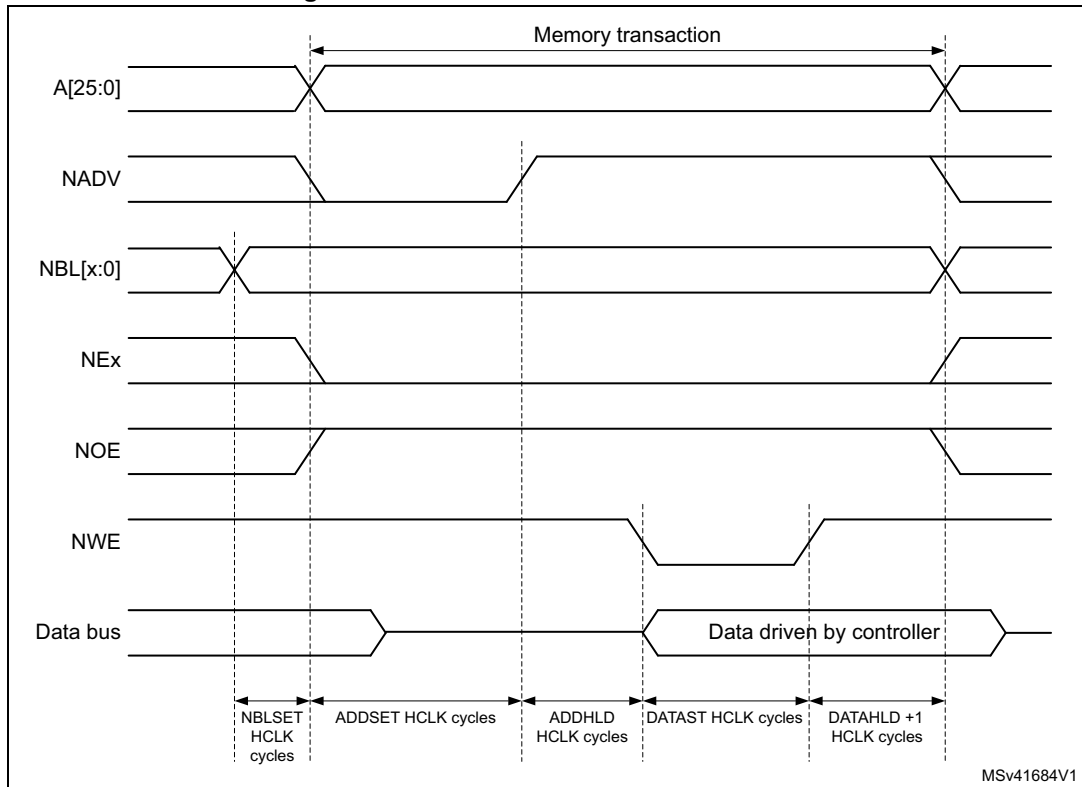




Figure 55. ModeD write access waveforms



The differences with Mode1 are the toggling of NOE that goes on toggling after NADV changes and the independent read and write timings.

Table 95. FSMC\_BCRx bit fields

Bit number	Bit name	Value to set
31:24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don't care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0

Table 95. FSMC\_BCRx bit fields (continued)

Bit number	Bit name	Value to set
7	Reserved	0x1
6	FACCEN	Set according to memory support
5:4	MWID	As needed
3:2	MTYP	As needed
1	MUXEN	0x0
0	MBKEN	0x1

Table 96. FSMC\_BTRx bit fields

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD HCLK cycles for read accesses).
29:28	ACCMOD	0x3
27:24	DATLAT	Don't care
23:20	CLKDIV	Don't care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for read accesses.
7:4	ADDHLD	Duration of the middle phase of the read access (ADDHLD HCLK cycles)
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 1.

Table 97. FSMC\_BWTRx bit fields

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD+1 HCLK cycles for write accesses).
29:28	ACCMOD	0x3
27:24	DATLAT	Don't care
23:20	CLKDIV	Don't care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles).
7:4	ADDHLD	Duration of the middle phase of the write access (ADDHLD HCLK cycles)
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 1.

Muxed mode - multiplexed asynchronous access to NOR Flash memory

Figure 56. Muxed read access waveforms

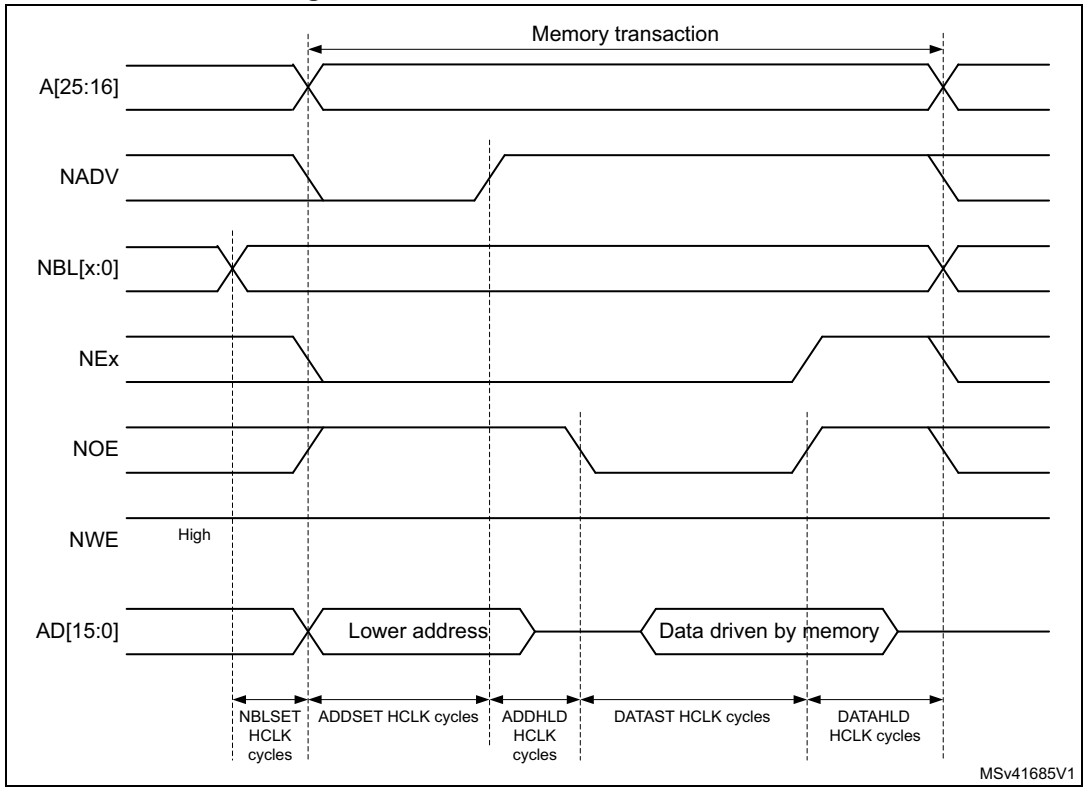
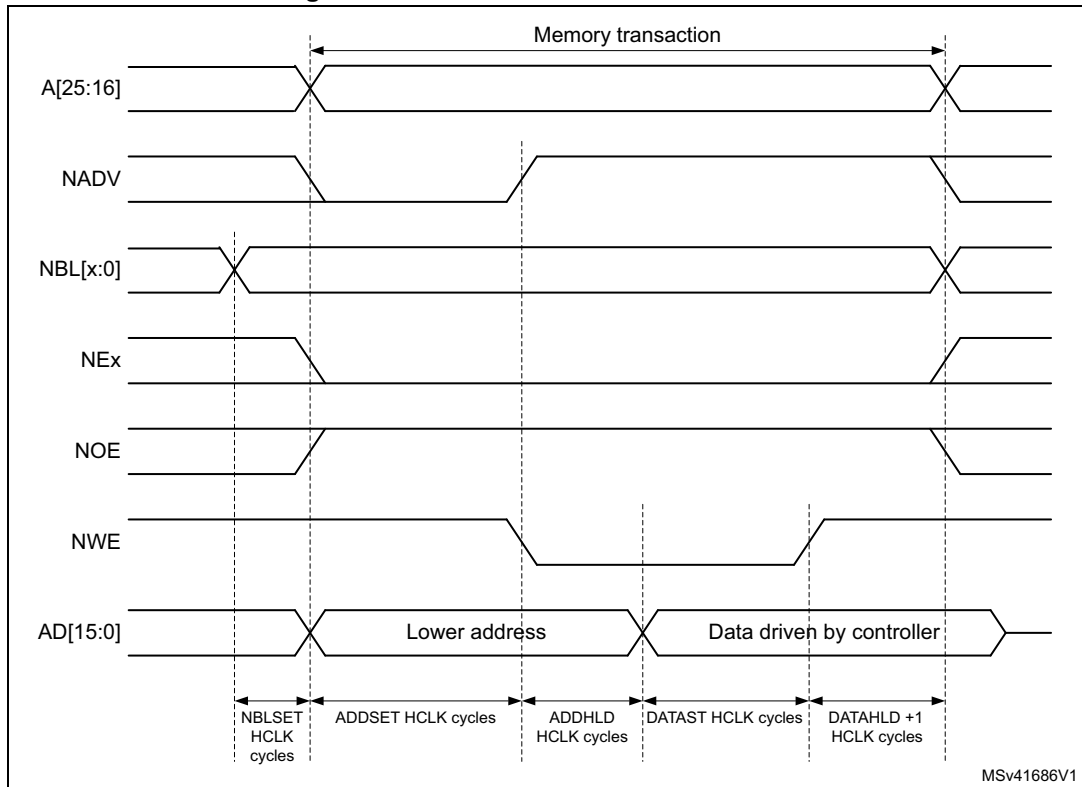


Figure 57. Muxed write access waveforms



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

Table 98. FSMC\_BCRx bit fields

Bit number	Bit name	Value to set
31:24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don't care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1

Table 98. FSMC\_BCRx bit fields (continued)

Bit number	Bit name	Value to set
6	FACCEN	0x1
5:4	MWID	As needed
3:2	MTYP	0x2 (NOR Flash memory) or 0x1(PSRAM)
1	MUXEN	0x1
0	MBKEN	0x1

Table 99. FSMC\_BTRx bit fields

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

### WAIT management in asynchronous accesses

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

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

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

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

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

$$\text{max\_wait\_assertion\_time} > \text{address\_phase} + \text{hold\_phase}$$

then:

$$\text{DATAST} \geq (4 \times \text{HCLK}) + (\text{max\_wait\_assertion\_time} - \text{address\_phase} - \text{hold\_phase})$$

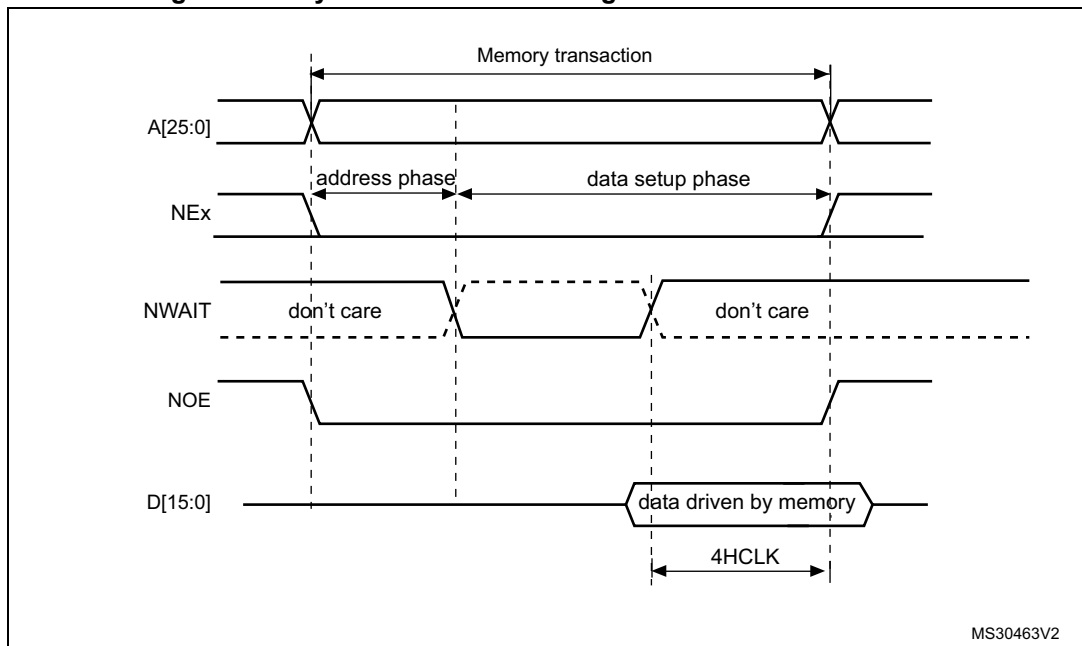
otherwise

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

where max\_wait\_assertion\_time is the maximum time taken by the memory to assert the WAIT signal once NEx/NOE/NWE is low.

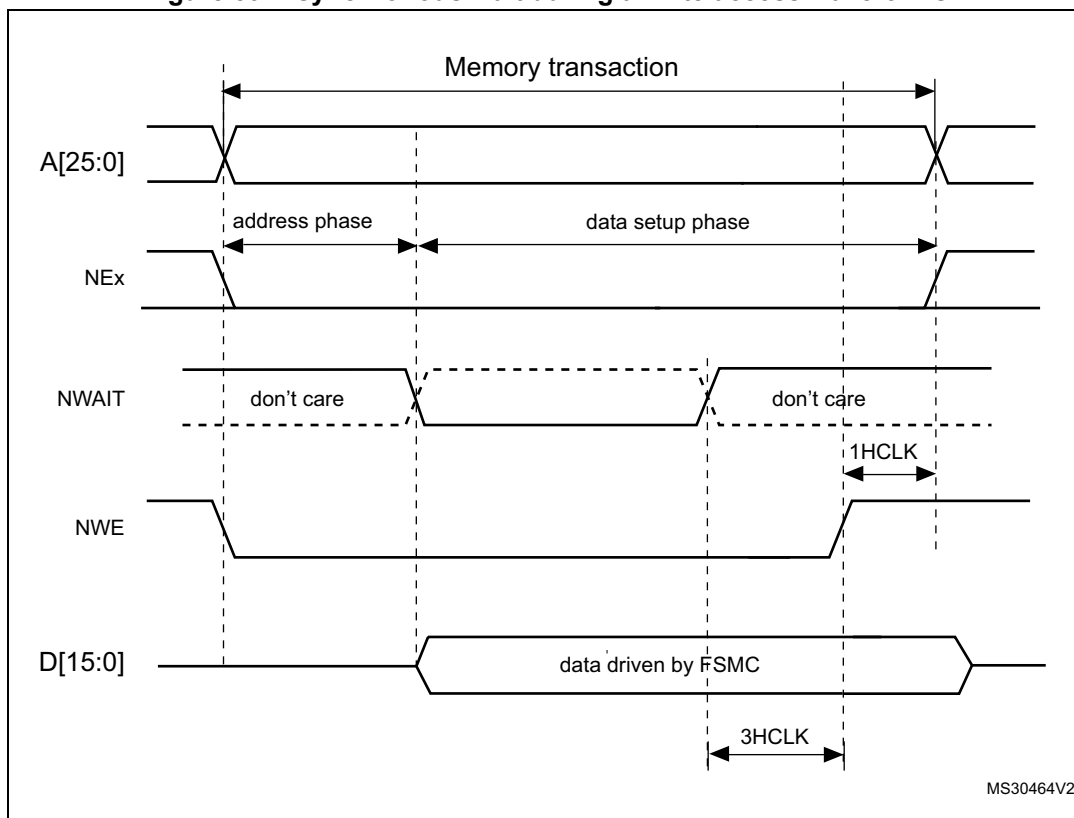
Figure 58 and Figure 59 show the number of HCLK clock cycles that are added to the memory access phase after WAIT is released by the asynchronous memory (independently of the above cases).

**Figure 58. Asynchronous wait during a read access waveforms**



1. NWAIT polarity depends on WAITPOL bit setting in FSMC\_BCRx register.

Figure 59. Asynchronous wait during a write access waveforms



1. NWAIT polarity depends on WAITPOL bit setting in FSMC\_BCRx register.

### 18.5.5 Synchronous transactions

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

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

Whatever MWID size: 16 or 8-bit, the FSMC\_CLK divider ratio is always defined by the programmed CLKDIV value.

Example:

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

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

#### Data latency versus NOR memory latency

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

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

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

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

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

### Single-burst transfer

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

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

### Cross boundary page for CellularRAM™ 1.5

CellularRAM™ 1.5 does not allow burst access to cross the page boundary. The FSMC controller allows to split automatically the burst access when the memory page size is reached by configuring the CPSIZE bits in the FSMC\_BCR1 register following the memory page size.

### Wait management

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

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

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

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

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

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

The FSMC supports both NOR Flash wait state configurations, for each chip select, thanks to the WAITCFG bit in the FSMC\_BCRx registers (x = 0..3).



Figure 60. Wait configuration waveforms

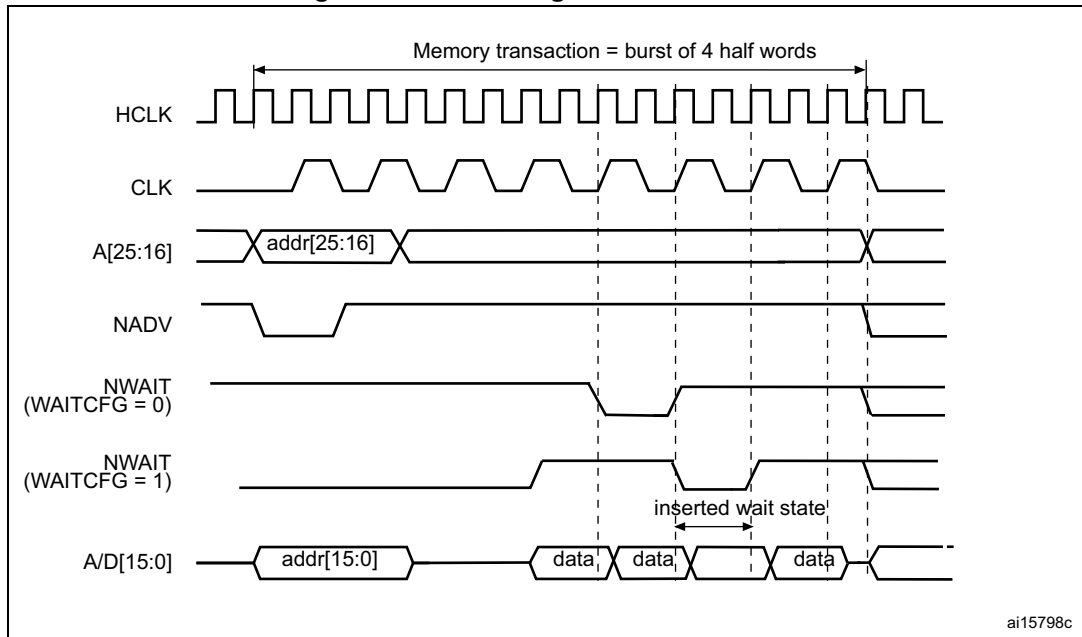
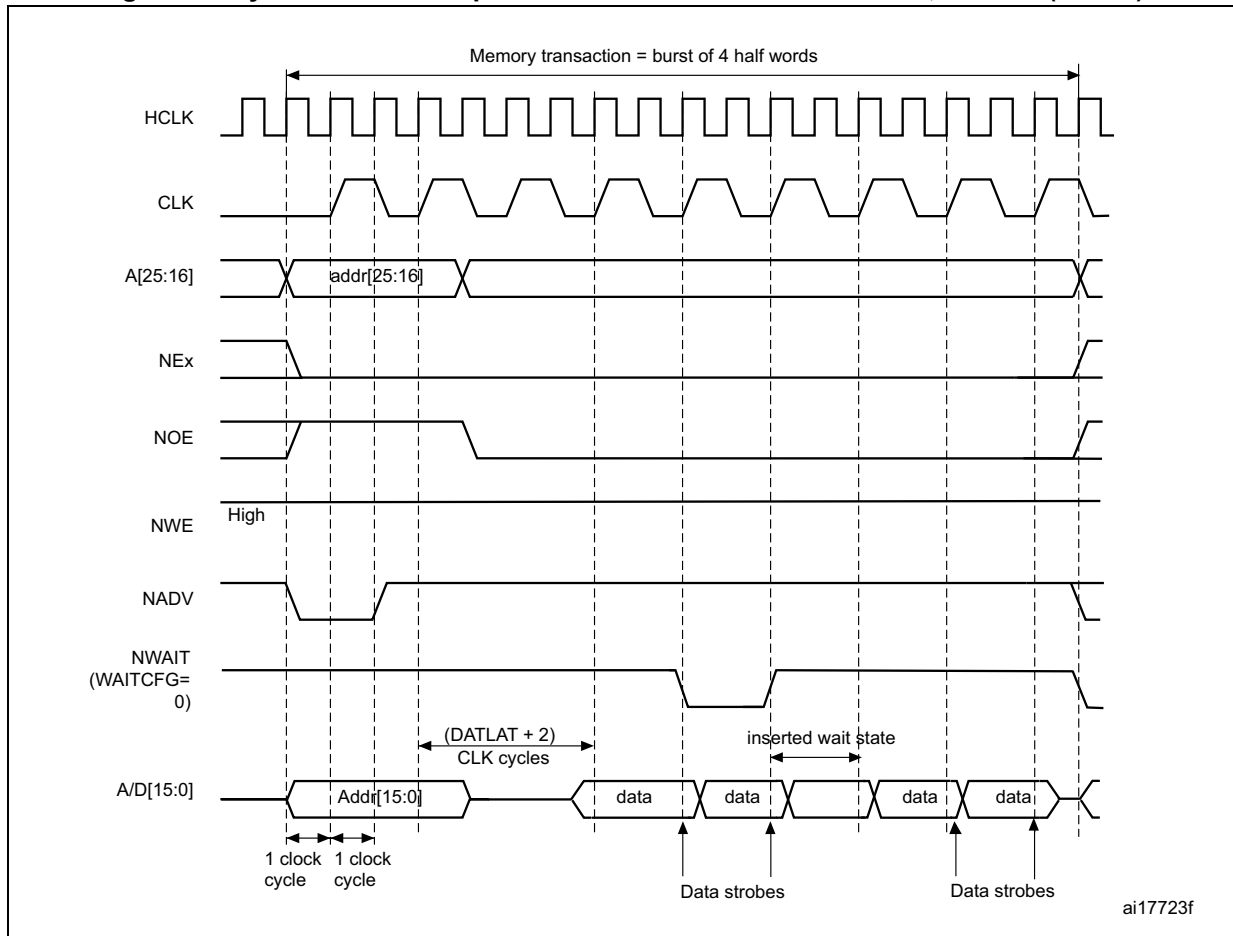


Figure 61. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)



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

Table 100. FSMC\_BCRx bit fields

Bit number	Bit name	Value to set
31-24	Reserved	0x000
23:22	NBLSET[1:0]	Don't care
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	No effect on synchronous read
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	To be set to 1 if the memory supports this feature, to be kept at 0 otherwise
12	WREN	No effect on synchronous read
11	WAITCFG	To be set according to memory

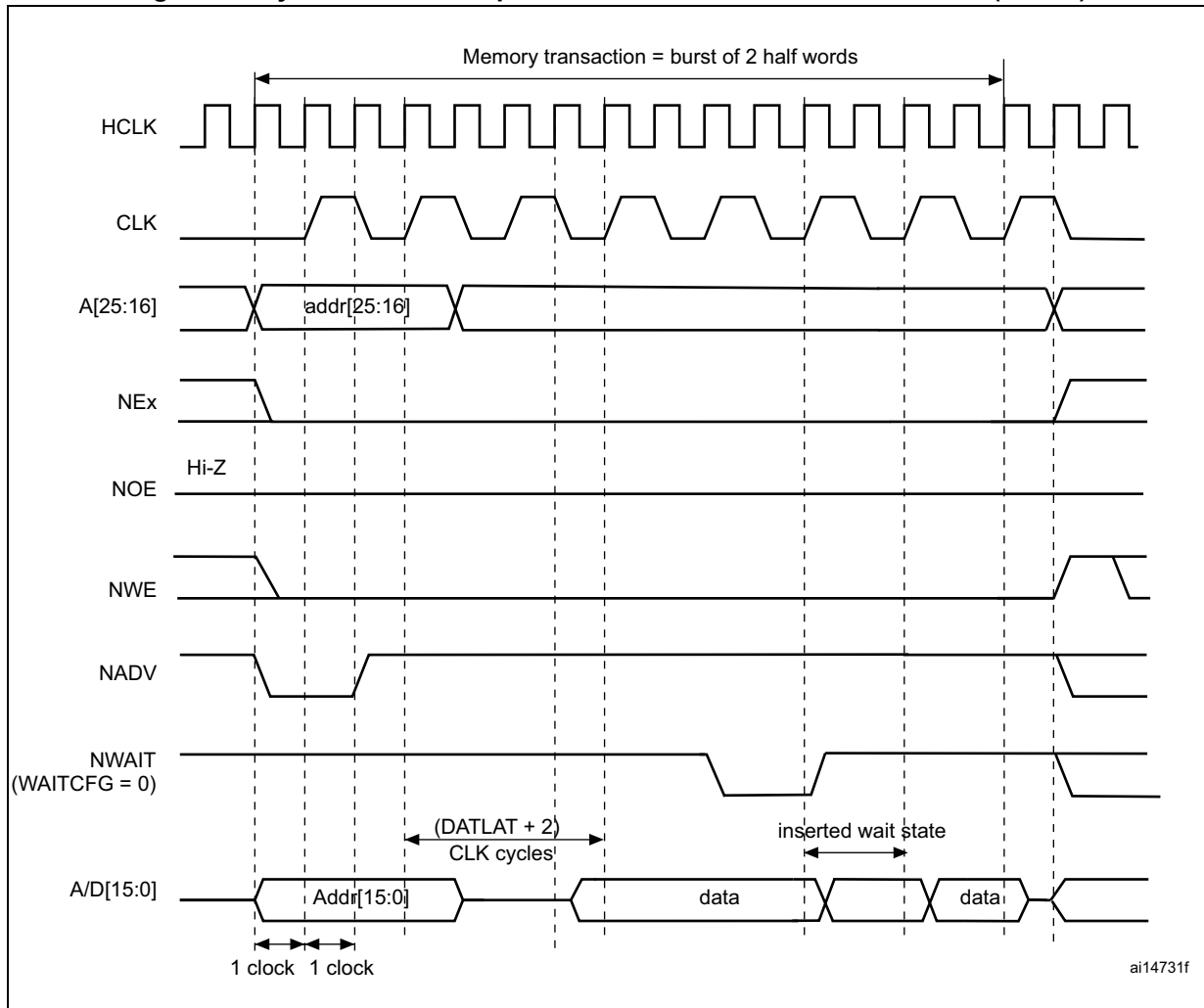
Table 100. FSMC\_BCRx bit fields (continued)

Bit number	Bit name	Value to set
10	Reserved	0x0
9	WAITPOL	To be set according to memory
8	BURSTEN	0x1
7	Reserved	0x1
6	FACCEN	Set according to memory support (NOR Flash memory)
5-4	MWID	As needed
3-2	MTYP	0x1 or 0x2
1	MUXEN	As needed
0	MBKEN	0x1

Table 101. FSMC\_BTRx bit fields

Bit number	Bit name	Value to set
31:30	DATAHLD	Don't care
29:28	ACCMOD	0x0
27-24	DATLAT	Data latency
27-24	DATLAT	Data latency
23-20	CLKDIV	0x0 to get CLK = HCLK 0x1 to get CLK = 2 × HCLK ..
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15-8	DATAST	Don't care
7-4	ADDHLD	Don't care
3-0	ADDSET	Don't care

Figure 62. Synchronous multiplexed write mode waveforms - PSRAM (CRAM)



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

Table 102. FSMC\_BCRx bit fields

Bit number	Bit name	Value to set
31-24	Reserved	0x000
23:22	NBLSET[1:0]	Don't care
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x1
18:16	CPSIZE	As needed (0x1 for CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0

Table 102. FSMC\_BCRx bit fields (continued)

Bit number	Bit name	Value to set
13	WAITEN	To be set to 1 if the memory supports this feature, to be kept at 0 otherwise.
12	WREN	0x1
11	WAITCFG	0x0
10	Reserved	0x0
9	WAITPOL	to be set according to memory
8	BURSTEN	no effect on synchronous write
7	Reserved	0x1
6	FACCEN	Set according to memory support
5-4	MWID	As needed
3-2	MTYP	0x1
1	MUXEN	As needed
0	MBKEN	0x1

Table 103. FSMC\_BTRx bit fields

Bit number	Bit name	Value to set
31-30	DATAHLD	Don't care
29:28	ACCMOD	0x0
27-24	DATLAT	Data latency
23-20	CLKDIV	0x0 to get CLK = HCLK 0x1 to get CLK = 2 × HCLK
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15-8	DATAST	Don't care
7-4	ADDHLD	Don't care
3-0	ADDSET	Don't care

### 18.5.6 NOR/PSRAM controller registers

#### SRAM/NOR-Flash chip-select control register for bank x (FSMC\_BCRx) (x = 1 to 4)

Address offset: 8 \* (x - 1), (x = 1 to 4)

Reset value: Bank 1: 0x0000 30DB

Reset value: Bank 2: 0x0000 30D2

Reset value: Bank 3: 0x0000 30D2

Reset value: Bank 4: 0x0000 30D2

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		WFDIS	CCLK EN	CBURST RW	CPSIZE[2:0]		
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASYNC WAIT	EXT MOD	WAIT EN	WREN	WAIT CFG	Res.	WAIT POL	BURST EN	Res.	FACC EN	MWID[1:0]		MTYP[1:0]		MUX EN	MBK EN
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 23:22 **NBLSET[1:0]**: Byte lane (NBL) SETUP

These bits configure the NBL setup timing from NBLx low to chip select NEx low.

- 00: NBL setup time is 0 AHB clock cycle
- 01: NBL setup time is 1 AHB clock cycle
- 10: NBL setup time is 2 AHB clock cycles
- 11: NBL setup time is 3 AHB clock cycles

Bit 21 **WFDIS**: Write FIFO Disable

This bit disables the Write FIFO used by the FSMC controller.

- 0: Write FIFO enabled (Default after reset)
- 1: Write FIFO disabled

*Note: The WFDIS bit of the FSMC\_BCR2..4 registers is don't care. It is only enabled through the FSMC\_BCR1 register.*

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

This bit enables the FSMC\_CLK clock output to external memory devices.

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

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

*Note: The CCLKEN bit of the FSMC\_BCR2..4 registers is don't care. It is only enabled through the FSMC\_BCR1 register. Bank 1 must be configured in synchronous mode to generate the FSMC\_CLK continuous clock.*

*Note: If CCLKEN bit is set, the FSMC\_CLK clock ratio is specified by CLKDIV value in the FSMC\_BTR1 register. CLKDIV in FSMC\_BWTR1 is don't care.*

*Note: If the synchronous mode is used and CCLKEN bit is set, the synchronous memories connected to other banks than Bank 1 are clocked by the same clock (the CLKDIV value in the FSMC\_BTR2..4 and FSMC\_BWTR2..4 registers for other banks has no effect.)*

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

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

0: Write operations are always performed in asynchronous mode

1: Write operations are performed in synchronous mode.

Bits 18:16 **CPSIZE[2:0]**: CRAM page size.

These are used for CellularRAM™ 1.5 which does not allow burst access to cross the address boundaries between pages. When these bits are configured, the FSMC controller splits automatically the burst access when the memory page size is reached (refer to memory datasheet for page size).

000: No burst split when crossing page boundary (default after reset)

001: 128 bytes

010: 256 bytes

011: 512 bytes

100: 1024 bytes

Others: reserved

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

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

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

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

Bit 14 **EXTMOD**: Extended mode enable.

This bit enables the FSMC to program the write timings for non multiplexed asynchronous accesses inside the FSMC\_BWTR register, thus resulting in different timings for read and write operations.

0: values inside FSMC\_BWTR register are not taken into account (default after reset)

1: values inside FSMC\_BWTR register are taken into account

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

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

- Bit 13 **WAITEN**: Wait enable bit.  
This bit enables/disables wait-state insertion via the NWAIT signal when accessing the memory in synchronous mode.  
0: NWAIT signal is disabled (its level not taken into account, no wait state inserted after the programmed Flash latency period)  
1: NWAIT signal is enabled (its level is taken into account after the programmed latency period to insert wait states if asserted) (default after reset)
- Bit 12 **WREN**: Write enable bit.  
This bit indicates whether write operations are enabled/disabled in the bank by the FSMC:  
0: Write operations are disabled in the bank by the FSMC, an AHB error is reported,  
1: Write operations are enabled for the bank by the FSMC (default after reset).
- Bit 11 **WAITCFG**: Wait timing configuration.  
The NWAIT signal indicates whether the data from the memory are valid or if a wait state must be inserted when accessing the memory in synchronous mode. This configuration bit determines if NWAIT is asserted by the memory one clock cycle before the wait state or during the wait state:  
0: NWAIT signal is active one data cycle before wait state (default after reset),  
1: NWAIT signal is active during wait state (not used for PSRAM).
- Bit 10 Reserved, must be kept at reset value.
- Bit 9 **WAITPOL**: Wait signal polarity bit.  
Defines the polarity of the wait signal from memory used for either in synchronous or asynchronous mode:  
0: NWAIT active low (default after reset),  
1: NWAIT active high.
- Bit 8 **BURSTEN**: Burst enable bit.  
This bit enables/disables synchronous accesses during read operations. It is valid only for synchronous memories operating in burst mode:  
0: Burst mode disabled (default after reset). Read accesses are performed in asynchronous mode.  
1: Burst mode enable. Read accesses are performed in synchronous mode.
- Bit 7 Reserved, must be kept at reset value.
- Bit 6 **FACCEN**: Flash access enable  
Enables NOR Flash memory access operations.  
0: Corresponding NOR Flash memory access is disabled  
1: Corresponding NOR Flash memory access is enabled (default after reset)
- Bits 5:4 **MWID[1:0]**: Memory data bus width.  
Defines the external memory device width, valid for all type of memories.  
00: 8 bits  
01: 16 bits (default after reset)  
10: reserved  
11: reserved



Bits 3:2 **MTYP[1:0]**: Memory type.

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

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

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

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

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

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

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

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

### SRAM/NOR-Flash chip-select timing register for bank x (FSMC\_BTRx)

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

Reset value: 0x0FFF FFFF

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

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

Bits 31:30 **DATAHLD[1:0]**: Data hold phase duration

These bits are written by software to define the duration of the data hold phase in HCLK cycles (refer to [Figure 45](#) to [Figure 57](#)), used in asynchronous accesses:

For read accesses

- 00: DATAHLD phase duration = 0 × HCLK clock cycle (default)
- 01: DATAHLD phase duration = 1 × HCLK clock cycle
- 10: DATAHLD phase duration = 2 × HCLK clock cycle
- 11: DATAHLD phase duration = 3 × HCLK clock cycle

For write accesses

- 00: DATAHLD phase duration = 1 × HCLK clock cycle (default)
- 01: DATAHLD phase duration = 2 × HCLK clock cycle
- 10: DATAHLD phase duration = 3 × HCLK clock cycle
- 11: DATAHLD phase duration = 4 × HCLK clock cycle

Bits 29:28 **ACCMOD[1:0]**: Access mode

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

00: access mode A

01: access mode B

10: access mode C

11: access mode D

Bits 27:24 **DATLAT[3:0]**: (see note below bit descriptions): Data latency for synchronous memory

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

This timing parameter is not expressed in HCLK periods, but in FSMC\_CLK periods.

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

0000: Data latency of 2 CLK clock cycles for first burst access

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

Bits 23:20 **CLKDIV[3:0]**: Clock divide ratio (for FSMC\_CLK signal)

Defines the period of FSMC\_CLK clock output signal, expressed in number of HCLK cycles:

0000: FSMC\_CLK period = 1x HCLK period

0001: FSMC\_CLK period = 2 × HCLK periods

0010: FSMC\_CLK period = 3 × HCLK periods

1111: FSMC\_CLK period = 16 × HCLK periods (default value after reset)

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

*Note: Refer to [Section 18.5.5: Synchronous transactions](#) for FSMC\_CLK divider ratio formula)*

Bits 19:16 **BUSTURN[3:0]**: Bus turnaround phase duration

These bits are written by software to add a delay at the end of current read or write transaction to next transaction on the same bank.

This delay allows to match the minimum time between consecutive transactions (tEHEL from NEx high to NEx low) and the maximum time needed by the memory to free the data bus after a read access (tEHQZ, Chip Enable high to output Hi-Z). This delay is recommended for mode D and muxed mode. For non-muxed memory, the bus turnaround delay can be set to minimum value.

$(BUSTURN + 1)HCLK\ period \geq \max(tEHEL\ min, tEHQZ\ max)$

For FRAM memories, the bus turnaround delay should be configured to match the minimum tPC (precharge time) timings. The bus turnaround delay is inserted between any consecutive transactions on the same bank (read/read, write/write, read/write and write/read) to match the tPC memory timing. The chip select is toggling between any consecutive accesses.

$(BUSTURN + 1)HCLK\ period \geq tPC\ min$

0000: BUSTURN phase duration = 1 HCLK clock cycle added

...

1111: BUSTURN phase duration = 16 x HCLK clock cycles added (default value after reset)

Bits 15:8 **DATAST[7:0]**: Data-phase duration

These bits are written by software to define the duration of the data phase (refer to [Figure 45](#) to [Figure 57](#)), used in asynchronous accesses:

0000 0000: Reserved

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

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

...

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

For each memory type and access mode data-phase duration, refer to the respective figure ([Figure 45](#) to [Figure 57](#)).

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

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

Bits 7:4 **ADDHLD[3:0]**: Address-hold phase duration

These bits are written by software to define the duration of the *address hold* phase (refer to [Figure 45](#) to [Figure 57](#)), used in mode D or multiplexed accesses:

0000: Reserved

0001: ADDHLD phase duration = 1 × HCLK clock cycle

0010: ADDHLD phase duration = 2 × HCLK clock cycle

...

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

For each access mode address-hold phase duration, refer to the respective figure ([Figure 45](#) to [Figure 57](#)).

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

Bits 3:0 **ADDSET[3:0]**: Address setup phase duration

These bits are written by software to define the duration of the *address setup* phase (refer to [Figure 45](#) to [Figure 57](#)), used in SRAMs, ROMs, asynchronous NOR Flash and PSRAM:

0000: ADDSET phase duration = 0 × HCLK clock cycle

...

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

For each access mode address setup phase duration, refer to the respective figure ([Figure 45](#) to [Figure 57](#)).

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

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

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

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

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

**SRAM/NOR-Flash write timing registers 1..4 (FSMC\_BWTR1..4)**

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

Reset value: 0x0FFF FFFF

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAHLD[1:0]		ACCMOD[1:0]		Res	Res	Res	Res	Res	Res	Res	Res	BUSTURN[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
DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **DATAHLD[1:0]**: Data hold phase duration

These bits are written by software to define the duration of the data hold phase in HCLK cycles (refer to [Figure 45](#) to [Figure 57](#)), used in asynchronous write accesses:

- 00: DATAHLD phase duration = 1 × HCLK clock cycle (default)
- 01: DATAHLD phase duration = 2 × HCLK clock cycle
- 10: DATAHLD phase duration = 3 × HCLK clock cycle
- 11: DATAHLD phase duration = 4 × HCLK clock cycle

Bits 29:28 **ACCMOD[1:0]**: Access mode.

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

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

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

Bits 19:16 **BUSTURN[3:0]**: Bus turnaround phase duration

These bits are written by software to add a delay at the end of current write transaction to next transaction on the same bank.

For FRAM memories, the bus turnaround delay should be configured to match the minimum tPC (precharge time) timings. The bus turnaround delay is inserted between any consecutive transactions on the same bank (read/read, write/write, read/write and write/read). The chip select is toggling between any consecutive accesses.

$$(BUSTURN + 1)HCLK \text{ period} \geq tPC \text{ min}$$

0000: BUSTURN phase duration = 1 HCLK clock cycle added

...

1111: BUSTURN phase duration = 16 x HCLK clock cycles added (default value after reset)

Bits 15:8 **DATAST[7:0]**: Data-phase duration.

These bits are written by software to define the duration of the data phase (refer to [Figure 45](#) to [Figure 57](#)), used in asynchronous SRAM, PSRAM and NOR Flash memory accesses:

0000 0000: Reserved

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

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

...

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

Bits 7:4 **ADDHLD[3:0]**: Address-hold phase duration.

These bits are written by software to define the duration of the *address hold* phase (refer to [Figure 54](#) to [Figure 57](#)), used in asynchronous multiplexed accesses:

0000: Reserved

0001: ADDHLD phase duration = 1 × HCLK clock cycle

0010: ADDHLD phase duration = 2 × HCLK clock cycle

...

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

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

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

These bits are written by software to define the duration of the *address setup* phase in HCLK cycles (refer to [Figure 45](#) to [Figure 57](#)), used in asynchronous accesses:

0000: ADDSET phase duration = 0 × HCLK clock cycle

...

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

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

## 18.6 NAND Flash controller

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

- 8- and 16-bit NAND Flash memories

The NAND bank is configured through dedicated registers ([Section 18.6.7](#)). The programmable memory parameters include access timings (shown in [Table 104](#)) and ECC configuration.

**Table 104. Programmable NAND Flash access parameters**

Parameter	Function	Access mode	Unit	Min.	Max.
Memory setup time	Number of clock cycles (HCLK) required to set up the address before the command assertion	Read/Write	AHB clock cycle (HCLK)	1	255
Memory wait	Minimum duration (in HCLK clock cycles) of the command assertion	Read/Write	AHB clock cycle (HCLK)	2	255

**Table 104. Programmable NAND Flash access parameters (continued)**

Parameter	Function	Access mode	Unit	Min.	Max.
Memory hold	Number of clock cycles (HCLK) during which the address must be held (as well as the data if a write access is performed) after the command de-assertion	Read/Write	AHB clock cycle (HCLK)	1	254
Memory databus high-Z	Number of clock cycles (HCLK) during which the data bus is kept in high-Z state after a write access has started	Write	AHB clock cycle (HCLK)	1	255

### 18.6.1 External memory interface signals

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

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

#### 8-bit NAND Flash memory

**Table 105. 8-bit NAND Flash**

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

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

**16-bit NAND Flash memory****Table 106. 16-bit NAND Flash**

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

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

### 18.6.2 NAND Flash supported memories and transactions

*Table 107 shows the supported devices, access modes and transactions. Transactions not allowed (or not supported) by the NAND Flash controller are shown in gray.*

**Table 107. Supported memories and transactions**

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

### 18.6.3 Timing diagrams for NAND Flash memory

The NAND Flash memory bank is managed through a set of registers:

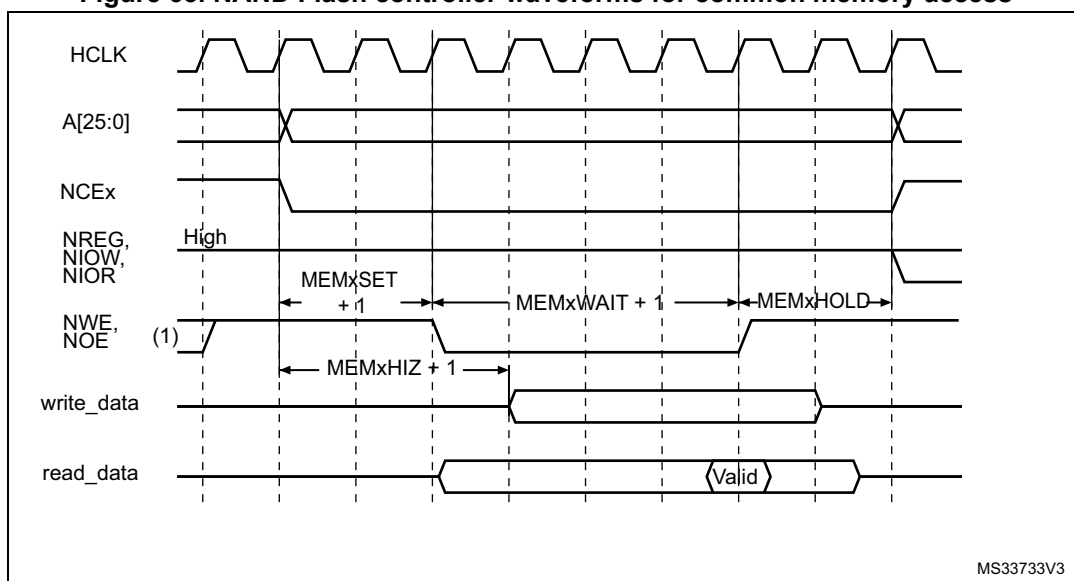
- Control register: FSMC\_PCR
- Interrupt status register: FSMC\_SR
- ECC register: FSMC\_ECCR
- Timing register for Common memory space: FSMC\_PMEM
- Timing register for Attribute memory space: FSMC\_PATT

Each timing configuration register contains three parameters used to define number of HCLK cycles for the three phases of any NAND Flash access, plus one parameter that defines the timing for starting driving the data bus when a write access is performed.

*Figure 63 shows the timing parameter definitions for common memory accesses, knowing that Attribute memory space access timings are similar.*



Figure 63. NAND Flash controller waveforms for common memory access



1. NOE remains high (inactive) during write accesses. NWE remains high (inactive) during read accesses.
2. For write access, the hold phase delay is (MEMHOLD) HCLK cycles and for read access is (MEMHOLD + 2) HCLK cycles.

### 18.6.4 NAND Flash operations

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

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

1. Program and enable the corresponding memory bank by configuring the FSMC\_PCR and FSMC\_PMEM (and for some devices, FSMC\_PATT, see [Section 18.6.5: NAND Flash prewait functionality](#)) registers according to the characteristics of the NAND Flash memory (PWID bits for the data bus width of the NAND Flash, PTYP = 1, PWAITEN = 0 or 1 as needed, see [Section 18.4.2: NAND Flash memory address mapping](#) for timing configuration).
2. The CPU performs a byte write to the common memory space, with data byte equal to one Flash command byte (for example 0x00 for Samsung NAND Flash devices). The LE input of the NAND Flash memory is active during the write strobe (low pulse on NWE), thus the written byte is interpreted as a command by the NAND Flash memory. Once the command is latched by the memory device, it does not need to be written again for the following page read operations.
3. The CPU can send the start address (STARTAD) for a read operation by writing four bytes (or three for smaller capacity devices), STARTAD[7:0], STARTAD[16:9], STARTAD[24:17] and finally STARTAD[25] (for 64 Mb x 8 bit NAND Flash memories) in the common memory or attribute space. The ALE input of the NAND Flash device is active during the write strobe (low pulse on NWE), thus the written bytes are interpreted as the start address for read operations. Using the attribute memory space makes it possible to use a different timing configuration of the FSMC, which can be

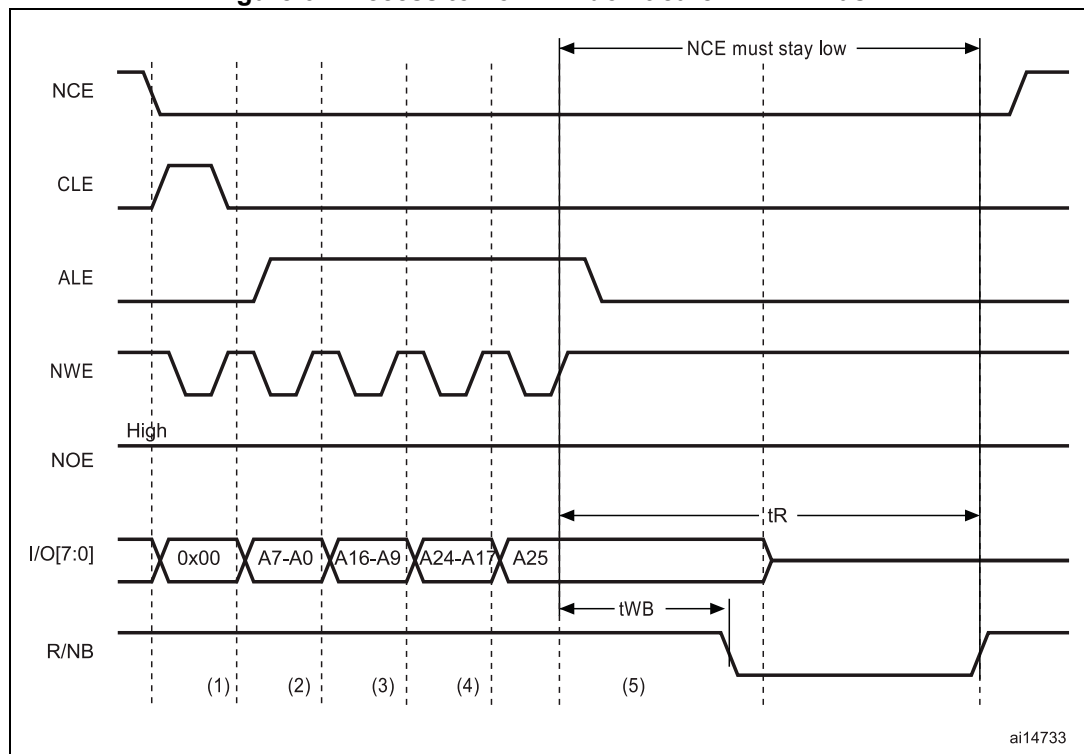
used to implement the prewait functionality needed by some NAND Flash memories (see details in [Section 18.6.5: NAND Flash prewait functionality](#)).

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

### 18.6.5 NAND Flash prewait functionality

Some NAND Flash devices require that, after writing the last part of the address, the controller waits for the R/NB signal to go low. (see [Figure 64](#)).

**Figure 64. Access to non ‘CE don’t care’ NAND-Flash**



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

When this functionality is required, it can be ensured by programming the MEMHOLD value to meet the  $t_{WB}$  timing. However any CPU read access to the NAND Flash memory has a hold delay of (MEMHOLD + 2) HCLK cycles and CPU write access has a hold delay of (MEMHOLD) HCLK cycles inserted between the rising edge of the NWE signal and the next access.

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

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

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

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

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

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

The ECC operates as follows:

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

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

To perform an ECC computation:

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

### 18.6.7 NAND Flash controller registers

#### NAND Flash control registers (FSMC\_PCR)

Address offset: 0x80

Reset value: 0x0000 0018

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCPS[2:0]			TAR3
												r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAR[2:0]			TCLR[3:0]				Res.	Res.	ECCEN	PWID[1:0]		PTYP	PBKEN	PWAITEN	Res.
r/w	r/w	r/w	r/w	r/w	r/w	r/w			r/w	r/w	r/w	r/w	r/w	r/w	

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

Bits 19:17 **ECCPS[2:0]**: ECC page size.

Defines the page size for the extended ECC:

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

Bits 16:13 **TAR[3:0]**: ALE to RE delay.

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

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

0000: 1 HCLK cycle (default)

1111: 16 HCLK cycles

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

Bits 12:9 **TCLR[3:0]**: CLE to RE delay.

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

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

0000: 1 HCLK cycle (default)

1111: 16 HCLK cycles

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

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

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

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

1: ECC logic is enabled.

Bits 5:4 **PWID[1:0]**: Data bus width.

Defines the external memory device width.

00: 8 bits

01: 16 bits (default after reset).

10: reserved.

11: reserved.

Bit 3 **PTYP**: Memory type.

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

0: Reserved, must be kept at reset value

1: NAND Flash (default after reset)

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

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

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

1: Corresponding memory bank is enabled

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

Enables the Wait feature for the NAND Flash memory bank:

0: disabled

1: enabled

Bit 0 Reserved, must be kept at reset value.

### FIFO status and interrupt register (FSMC\_SR)

Address offset: 0x84

Reset value: 0x0000 0040

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

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS
									r	rw	rw	rw	rw	rw	rw

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

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

**Common memory space timing register 2..4 (FSMC\_PMEM)**

Address offset: Address: 0x88

Reset value: 0xFCFC FCFC

The FSMC\_PMEM read/write register contains the timing information for NAND Flash memory bank. This information is used to access either the common memory space of the NAND Flash for command, address write access and data read/write access.



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

Bits 31:24 **MEMHIZ[7:0]**: Common memory x data bus Hi-Z time  
 Defines the number of HCLK clock cycles during which the data bus is kept Hi-Z after the start of a NAND Flash write access to common memory space on socket. This is only valid for write transactions:  
 0000 0000: 1 HCLK cycle  
 1111 1110: 255 HCLK cycles  
 1111 1111: reserved.

Bits 23:16 **MEMHOLD[7:0]**: Common memory hold time  
 Defines the number of HCLK clock cycles for write access and HCLK (+2) clock cycles for read access during which the address is held (and data for write accesses) after the command is deasserted (NWE, NOE), for NAND Flash read or write access to common memory space on socket x:  
 0000 0000: reserved.  
 0000 0001: 1 HCLK cycle for write access / 3 HCLK cycles for read access  
 1111 1110: 254 HCLK cycles for write access / 256 HCLK cycles for read access  
 1111 1111: reserved.

Bits 15:8 **MEMWAIT[7:0]**: Common memory wait time  
 Defines the minimum number of HCLK (+1) clock cycles to assert the command (NWE, NOE), for NAND Flash read or write access to common memory space on socket. The duration of command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of HCLK:  
 0000 0000: reserved  
 0000 0001: 2HCLK cycles (+ wait cycle introduced by deasserting NWAIT)  
 1111 1110: 255 HCLK cycles (+ wait cycle introduced by deasserting NWAIT)  
 1111 1111: reserved.

Bits 7:0 **MEMSET[7:0]**: Common memory x setup time  
 Defines the number of HCLK (+1) clock cycles to set up the address before the command assertion (NWE, NOE), for NAND Flash read or write access to common memory space on socket x:  
 0000 0000: 1 HCLK cycle  
 1111 1110: 255 HCLK cycles  
 1111 1111: reserved

**Attribute memory space timing registers (FSMC\_PATT)**

Address offset: 0x8C

Reset value: 0xFCFC FCFC

The FSMC\_PATT read/write register contains the timing information for NAND Flash memory bank. It is used for 8-bit accesses to the attribute memory space of the NAND Flash for the last address write access if the timing must differ from that of previous accesses (for Ready/Busy management, refer to [Section 18.6.5: NAND Flash prewait functionality](#)).

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

Bits 31:24 **ATTTHIZ[7:0]**: Attribute memory data bus Hi-Z time

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

- 0000 0000: 0 HCLK cycle
- 1111 1110: 255 HCLK cycles
- 1111 1111: reserved.

Bits 23:16 **ATTTHOLD[7:0]**: Attribute memory hold time

Defines the number of HCLK clock cycles for write access and HCLK (+2) clock cycles for read access during which the address is held (and data for write access) after the command deassertion (NWE, NOE), for NAND Flash read or write access to attribute memory space on socket:

- 0000 0000: reserved
- 0000 0001: 1 HCLK cycle for write access / 3 HCLK cycles for read access
- 1111 1110: 254 HCLK cycles for write access / 256 HCLK cycles for read access
- 1111 1111: reserved.

Bits 15:8 **ATTWAIT[7:0]**: Attribute memory wait time

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

- 0000 0000: reserved
- 0000 0001: 2 HCLK cycles (+ wait cycle introduced by deassertion of NWAIT)
- 1111 1110: 255 HCLK cycles (+ wait cycle introduced by deasserting NWAIT)
- 1111 1111: reserved.

Bits 7:0 **ATTSET[7:0]**: Attribute memory setup time

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

- 0000 0000: 1 HCLK cycle
- 1111 1110: 255 HCLK cycles
- 1111 1111: reserved.

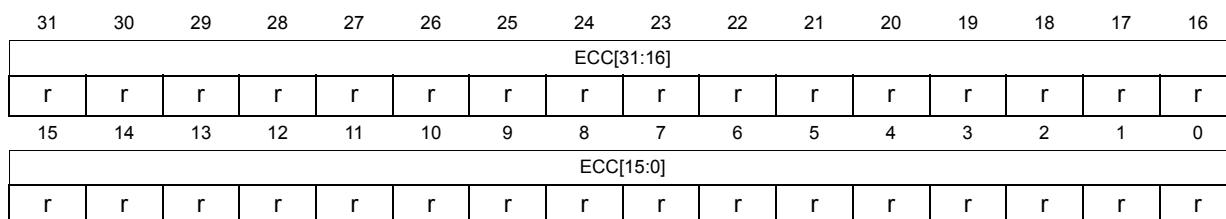


### ECC result registers (FSMC\_ECCR)

Address offset: 0x94

Reset value: 0x0000 0000

This register contain the current error correction code value computed by the ECC computation modules of the FSMC NAND controller. When the CPU reads the data from a NAND Flash memory page at the correct address (refer to [Section 18.6.6: Computation of the error correction code \(ECC\) in NAND Flash memory](#)), the data read/written from/to the NAND Flash memory are processed automatically by the ECC computation module. When X bytes have been read (according to the ECCPS field in the FSMC\_PCR registers), the CPU must read the computed ECC value from the FSMC\_ECC registers. It then verifies if these computed parity data are the same as the parity value recorded in the spare area, to determine whether a page is valid, and, to correct it otherwise. The FSMC\_ECCR register should be cleared after being read by setting the ECCEN bit to '0'. To compute a new data block, the ECCEN bit must be set to '1'.



Bits 31:0 **ECC[31:0]**: ECC result

This field contains the value computed by the ECC computation logic. [Table 108](#) describes the contents of these bit fields.

**Table 108. ECC result relevant bits**

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

## 18.7 FSMC register map

Table 109. FSMC register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FSMC_BCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBL SET [1:0]		WFDIS	CCLKEN	CBURSTRW	CPSIZE [2:0]			ASYNCAWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID [1:0]		MTYP [1:0]	MUXEN	MBKEN	
	Reset value									0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	1	1
0x08	FSMC_BCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBL SET [1:0]		Res.	Res.	CBURSTRW	CPSIZE [2:0]			ASYNCAWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID [1:0]		MTYP [1:0]	MUXEN	MBKEN	
	Reset value									0	0			0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	0	1	0
0x10	FSMC_BCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBL SET [1:0]		Res.	Res.	CBURSTRW	CPSIZE [2:0]			ASYNCAWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID [1:0]		MTYP [1:0]	MUXEN	MBKEN	
	Reset value									0	0			0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	0	1	0
0x18	FSMC_BCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBL SET [1:0]		Res.	Res.	CBURSTRW	CPSIZE [2:0]			ASYNCAWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID [1:0]		MTYP [1:0]	MUXEN	MBKEN	
	Reset value									0	0			0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	0	1	0
0x04	FSMC_BTR1	DATAHLD [1:0]		ACCMOD[1:0]		DATLAT[3:0]			CLKDIV[3:0]			BUSTURN[3:0]			DATAST[7:0]							ADDHLD[3:0]			ADDSET[3:0]								
	Reset value	0	0	0	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
0x0C	FSMC_BTR2	DATAHLD [1:0]		ACCMOD[1:0]		DATLAT[3:0]			CLKDIV[3:0]			BUSTURN[3:0]			DATAST[7:0]							ADDHLD[3:0]			ADDSET[3:0]								
	Reset value	0	0	0	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
0x14	FSMC_BTR3	DATAHLD [1:0]		ACCMOD[1:0]		DATLAT[3:0]			CLKDIV[3:0]			BUSTURN[3:0]			DATAST[7:0]							ADDHLD[3:0]			ADDSET[3:0]								
	Reset value	0	0	0	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
0x1C	FSMC_BTR4	DATAHLD [1:0]		ACCMOD[1:0]		DATLAT[3:0]			CLKDIV[3:0]			BUSTURN[3:0]			DATAST[7:0]							ADDHLD[3:0]			ADDSET[3:0]								
	Reset value	0	0	0	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
0x104	FSMC_BWTR1	DATAHLD [1:0]		ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		BUSTURN[3:0]			DATAST[7:0]							ADDHLD[3:0]			ADDSET[3:0]					
	Reset value	0	0	0	0										1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 109. FSMC register map (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10C	FSMC_BWTR2	DATAHLD [1:0]		ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN[3:0]			DATAST[7:0]							ADDHLD[3:0]			ADDSET[3:0]						
	Reset value	0	0	0	0										1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x114	FSMC_BWTR3	DATAHLD [1:0]		ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN[3:0]			DATAST[7:0]							ADDHLD[3:0]			ADDSET[3:0]						
	Reset value	0	0	0	0										1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x11C	FSMC_BWTR4	DATAHLD [1:0]		ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN[3:0]			DATAST[7:0]							ADDHLD[3:0]			ADDSET[3:0]						
	Reset value	0	0	0	0										1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x80	FSMC_PCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCPS [2:0]		TAR[3:0]			TCLR[3:0]			Res.	Res.	Res.	Res.	ECCEN	PWID [1:0]		PTYP	PBKEN	PWAITEN	Res.	
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0x84	FSMC_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.	FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS
	Reset value																										1	0	0	0	0	0	0
0x88	FSMC_PMEM	MEMHIZx[7:0]							MEMHOLDx[7:0]							MEMWAITx[7:0]							MEMSETx[7:0]										
	Reset value	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0
0x8C	FSMC_PATT	ATTHIZ[7:0]							ATTHOLD[7:0]							ATTWAIT[7:0]							ATTSET[7:0]										
	Reset value	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0
0x94	FSMC_ECCR	ECCx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.

## 19 Octo-SPI interface (OCTOSPI)

The OCTOSPI is a specialized communication interface targeting single, dual, quad or octal SPI memories. It can operate in any of the three following modes:

- Indirect mode: all the operations are performed using the OCTOSPI registers
- Status-polling mode: the external memory status register is periodically read and an interrupt can be generated in case of flag setting
- Memory-mapped mode: the external memory is memory mapped and it is seen by the system as if it was an internal memory supporting both read and write operations.

The OCTOSPI supports two frame formats:

- The classical frame format with the command, address, alternate byte, dummy cycles and data phase
- The HyperBus™ frame format.

### 19.1 OCTOSPI main features

- Three functional modes: indirect, status-polling, and memory-mapped
- Read and write support in memory-mapped mode
- Supports for single, dual, quad and octal communication
- Dual-quad mode, where 8 bits can be sent/received simultaneously by accessing two quad memories in parallel
- SDR and DTR support
- Data strobe support
- Fully programmable opcode
- Fully programmable frame format
- HyperBus™ support
- Integrated FIFO for reception and transmission
- 8, 16, and 32-bit data accesses are allowed
- DMA channel for indirect mode operations
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error.

## 19.2 OCTOSPI functional description

Figure 65. OCTOSPI block diagram for octal communication

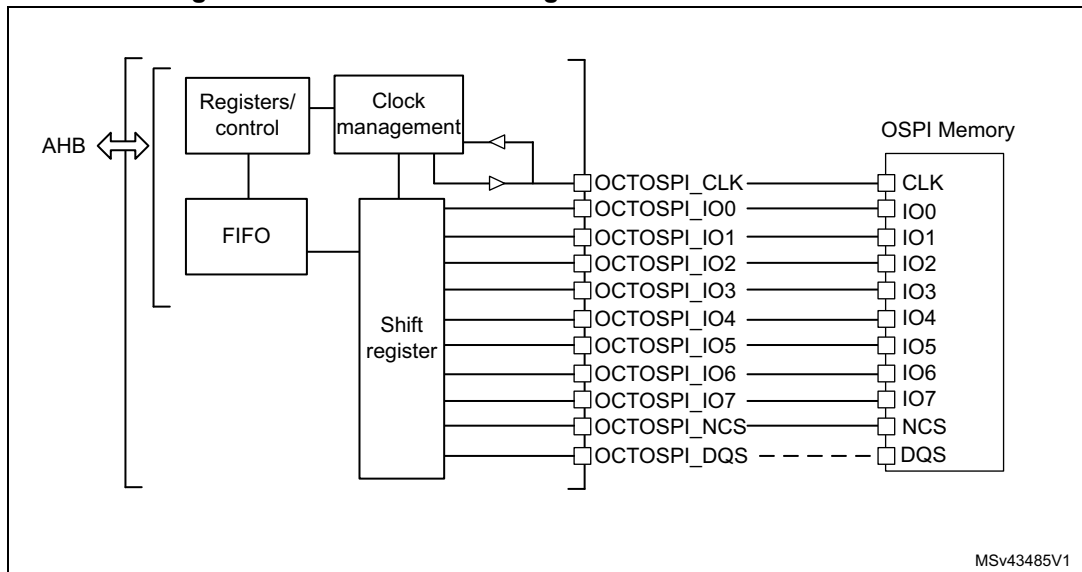


Figure 66. OCTOSPI block diagram in quad mode

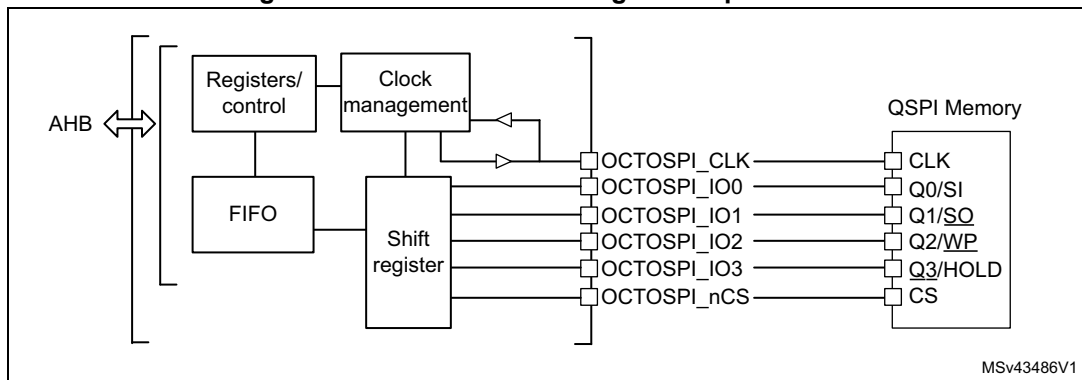
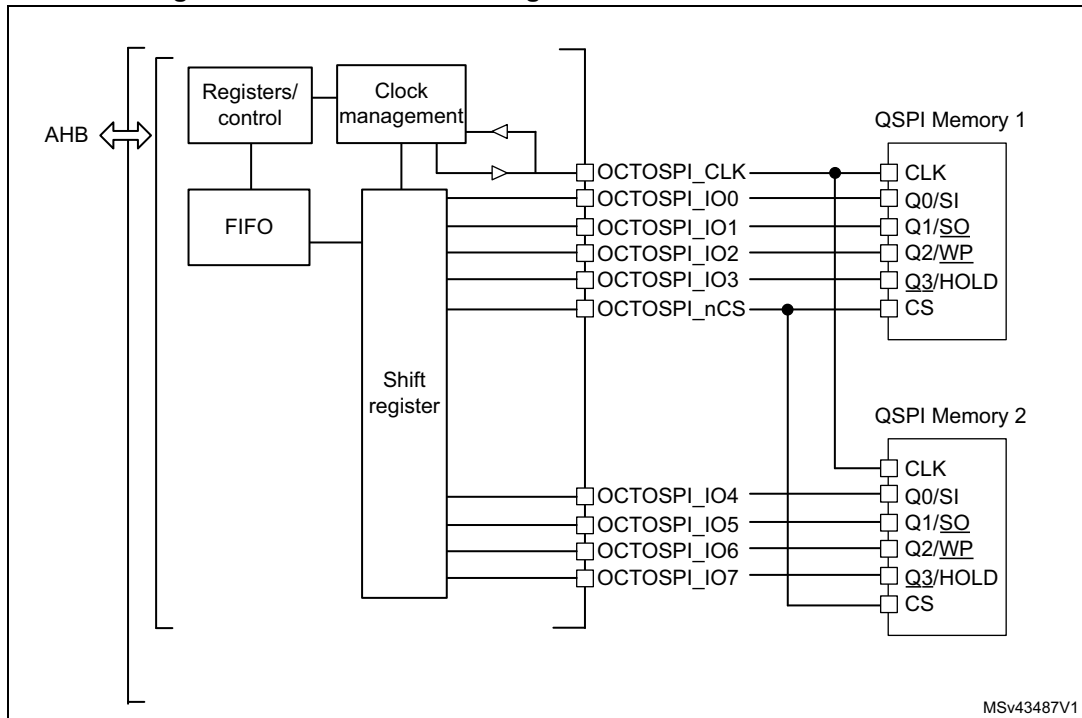


Figure 67. OCTOSPI block diagram when dual-Flash is enabled



MSv43487V1

The OCTOSPI can operate in two different operating modes:

- Regular-command mode
- HyperBus™ mode

In any of these two modes, the OCTOSPI uses from 6 to up to 12 signals to interface with a memory, depending on the functional mode:

- nCS: chip-select used in all modes
- CLK: communication clock used in all modes
- DQS: data strobe used mainly in octal mode
- IO[3:0]: data bus LSB used in all modes
- IO[7:4]: data bus MSB used in dual-quad and octal modes.

### 19.2.1 OCTOSPI regular-command mode

When in regular-command mode, the OCTOSPI communicates with the external device using commands. Each command can include 5 phases:

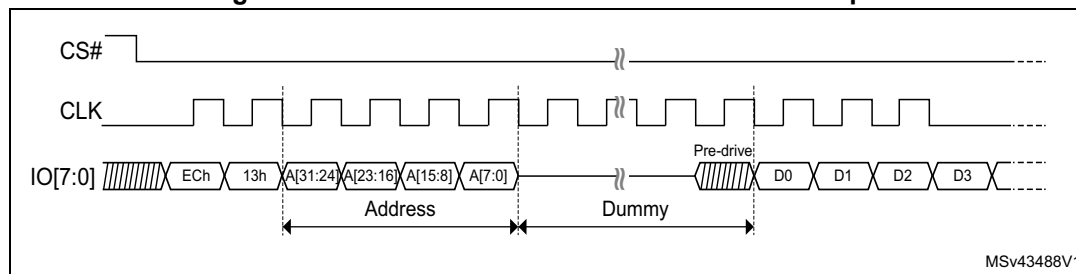
- Instruction phase
- Address phase
- Alternate-byte phase
- Dummy phase
- Data phase.

Any of these phases can be configured to be skipped, but at least one of the instruction, address, alternate byte, or data phases must be present.

The nCS falls before the start of each command and rises again after each command finishes.

In memory-mapped mode, both read and write operation are supported, as a consequence, some of the configuration registers are duplicated to specify write operations (read operations are configured using regular registers).

**Figure 68. SDR read command in octal mode example**



The specific regular-command mode features are configured through the registers in the 0x0100-0x01FC offset range.

### Instruction phase

During this phase, a 1 to 4 bytes instruction is sent to the external device specifying the type of operation to be performed. The size of the instruction to be sent is configured in the ISIZE[1:0] (OCTOSPI\_CCR[5:4]) field and the instruction is programmed in the OCTOSPI\_IR register's INSTRUCTION[31:0] field.

Though most devices 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), 4-bits at a time (over IO0 to IO3 in Quad-SPI mode) or 8-bits at a time (over IO0 to IO7 in octal SPI mode). This can be configured using the IMODE[2:0] (OCTOSPI\_CCR[2:0]) field.

The instruction can be sent in DRT (double transfer rate) mode on each rising and falling edge of the clock by setting the IDTR (OCTOSPI\_CCR[3]) bit.

When the IMODE=000, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

When in memory-mapped mode, the instruction used for the write operation is specified in the OCTOSPI\_WIR register and the instruction format is specified in the OCTOSPI\_WCCR register. The instruction used for the read operation and the instruction format are specified in the regular register (OCTOSPI\_IR and OCTOSPI\_CCR).

### Address phase

In the address phase, 1 to 4 bytes are sent to the external device to indicate the address of the operation. The number of the address bytes to be sent is configured in the ADSIZE[1:0] (OCTOSPI\_CCR[13:12]) field.

In the indirect and automatic-polling modes, the address bytes to be sent are specified in the ADDRESS[31:0] (OCTOSPI\_AR) field, while in the memory-mapped mode the address is given directly via the AHB (from the Cortex<sup>®</sup>-M Core 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), 4-bits at a time (over IO0 to IO3 in Quad-SPI mode) or 8-bits at a time (over IO0 to IO7 in octal SPI mode). This can be configured using the ADMODE[2:0] (OCTOSPI\_CCR[10:8]) field.

The address can be sent in DTR mode (on each rising and falling edge of the clock) setting the ADDTR (OCTOSPI\_CCR[11]) bit.

When ADMODE=000, the address phase is skipped, and the command sequence proceeds directly to the next phase, if any.

When in memory-mapped mode, the address format for the write operation is specified in the OCTOSPI\_WCCR register. The address format for the read operation is specified in the regular register (OCTOSPI\_CCR).

### Alternate-bytes phase

In the alternate-bytes phase, 1 to 4 bytes are sent to the external device, generally to control the mode of operation. The number of alternate bytes to be sent is configured in the ABSIZE[1:0] (OCTOSPI\_CCR[21:20]) field. The bytes to be sent are specified in the OCTOSPI\_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 and IO1 in dual-SPI mode), 4-bits at a time (over IO0 to IO3 in Quad-SPI mode) or 8-bits at a time (over IO0 to IO7 in octal SPI mode). This can be configured using the ABMODE[2:0] (OCTOSPI\_CCR[18:16]) field.

The alternate bytes can be sent in DTR mode (on each rising and falling edge of the clock) setting the ABDTR (OCTOSPI\_CCR[19]) bit.

When ABMODE=000, the alternate-bytes 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 the Quad-SPI mode (ABMODE=011) 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 should be set to 0x8A (1000\_1010).

When in memory-mapped mode, the alternate bytes used for the write operation are specified in the OCTOSPI\_WABR register and the alternate byte format is specified in the OCTOSPI\_WCCR register. The alternate bytes used for read operation and the alternate byte format are specified in the regular register (OCTOSPI\_ABR and OCTOSPI\_CCR)

### Dummy-cycles phase

In the dummy-cycles phase, 1 to 31 cycles are given without any data being sent or received, in order to give the external device the time to prepare for the data phase when the higher clock frequencies are used. The number of cycles given during this phase is specified in the DCYC[4:0] (OCTOSPI\_TCR[4:0]) field. In both SDR and DTR 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.

In order to assure enough "turn-around" time for changing the data signals from the output mode to the input mode, there must be at least one dummy cycle when using the dual, the quad or the octal mode to receive data from the external device.



It is recommended to have at least five dummy cycles when using memories with DQS activated.

When in memory-mapped mode, the dummy cycles for the write operations are specified in the OCTOSPI\_WTCR register. The dummy cycles for the read operation are specified in the regular register (OCTOSPI\_TCR)

### Data phase

During the data phase, any number of bytes can be sent to or received from the external device.

When in indirect mode, the number of bytes to be sent/received is specified in the OCTOSPI\_DLR register.

When in indirect-write mode the data to be sent to the external device must be written to the OCTOSPI\_DR register, while in indirect-read mode the data received from the external device is obtained by reading from the OCTOSPI\_DR register.

In automatic-polling modes, the number of bytes to be received is specified in the OCTOSPI\_DLR register and the data received from the external device can be obtained by reading from the OCTOSPI\_DR register.

When in memory-mapped mode, the data which is read or written is sent or received directly over the AHB to the Cortex<sup>®</sup> 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), 4-bits at a time (over IO0 to IO3 in Quad-SPI mode) or 8-bits at a time (over IO0 to IO7 in octal SPI mode). This can be configured using the DMODE[2:0] (OCTOSPI\_CCR[26:24]) field.

The data can be sent or received in DTR mode (on each rising and falling edge of the clock) setting the DDTR (OCTOSPI\_CCR[27]) bit.

When DMODE=000, the data phase is skipped, and the command sequence finishes immediately by raising the nCS. This configuration must be used only in indirect-write mode.

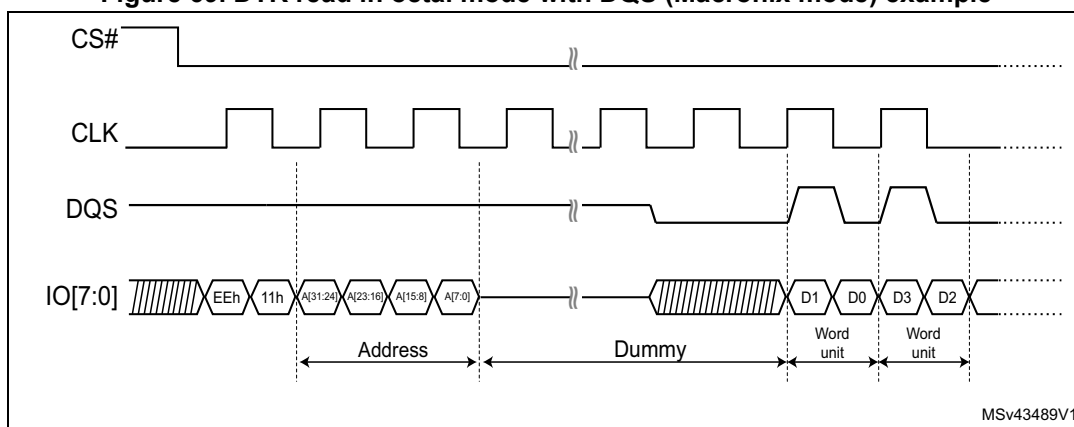
When in memory-mapped mode, the data format for the write operation is specified in the OCTOSPI\_WCCR register. The data format for the read operation is specified in the regular register (OCTOSPI\_CCR).

### DQS usage

The DQS signal can be used for data strobing during the read transactions when the device is toggling the DQS aligned with the data.

The DQS management can be enabled setting the DQS enable (DQSE) bit of the OCTOSPI communication configuration register (OCTOSPI\_CCR).

Figure 69. DTR read in octal mode with DQS (Macronix mode) example



MSv43489V1

## 19.2.2 OCTOSPI regular-command mode signal interface

### Single-SPI mode

The legacy SPI mode allows just a single bit to be sent/received serially. In this mode, the data is sent to the external device over the SO signal (whose I/O are shared with IO0). The data received from the external device arrives via SI (whose I/O are shared with IO1).

The different phases can each be configured separately to use this single-bit mode by setting the IMODE/ADMODE/ABMODE/DMODE fields (in OCTOSPI\_CCR and OCTOSPI\_WCCR) to 001.

In each phase 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' (to deactivate the "write protect" function)
- IO3 is in output mode and forced to '1' (to deactivate the "hold" function)
- IO4/IO5/IO6/IO7 are in output mode and forced to '0'

This is the case even for the dummy phase if DMODE=001.

### 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 (in OCTOSPI\_CCR and OCTOSPI\_WCCR) to 010.

In each phase which is configured in dual-SPI mode:

- IO0/IO1 are at high-impedance (input) during the data phase for the 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'
- IO4/IO5/IO6/IO7 are in output mode and forced to '0'

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 the Quad-SPI mode by setting the IMODE/ADMODE/ABMODE/DMODE fields (in OCTOSPI\_CCR and OCTOSPI\_WCCR) to 011.

In each phase which is configured in Quad-SPI mode, IO0/IO1/IO2/IO3 are all at high-impedance (input) during the data phase for the read operations, and outputs in all other cases. IO4/IO5/IO6/IO7 are in output mode and forced to '0

In the dummy phase when DMODE=011, 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 the Quad-SPI mode, then the pins corresponding to IO2 and IO3 can be used for other functions even while the OCTOSPI is active.

### Octo-SPI mode

In regular Octo-SPI mode, the eight bits are sent/received simultaneously over the IO[0:7] signals.

The different phases can each be configured separately to use the Octo-SPI mode by setting the IMODE/ADMODE/ABMODE/DMODE fields (in OCTOSPI\_CCR and OCTOSPI\_WCCR) to 100.

In each phase that is configured in octal mode, IO[0:7] are all at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase when DMODE=100, IO[0:7] are all high-impedance.

IO[4:7] are used only in Octo-SPI mode. If none of the phases are configured to use Octo-SPI mode, then the pins corresponding to IO[4:7] can be used for other functions even while the OCTOSPI is active.

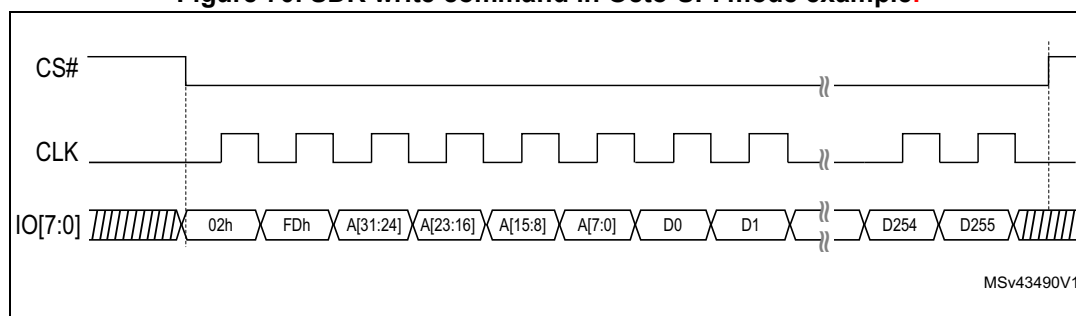
### SDR mode

By default, all the phases operate in single data rate (SDR) mode.

In SDR mode, when the OCTOSPI is driving the IO0/SO, IO1, IO2, IO3, IO4, IO5, IO6 and IO7 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the OCTOSPI assumes that the external devices also send the data using CLK's falling edge. By default (when SSHIFT=0), the signals are sampled using the following (rising) edge of CLK:

Figure 70. SDR write command in Octo-SPI mode example.



### DTR mode

Each of the instruction/address/alternate/data phase can be configured to operate in double transfer rate (DTR) mode setting IDTR/ADDTR/ABDTR/DDTR bit in the OCTOSPI\_CCR register.

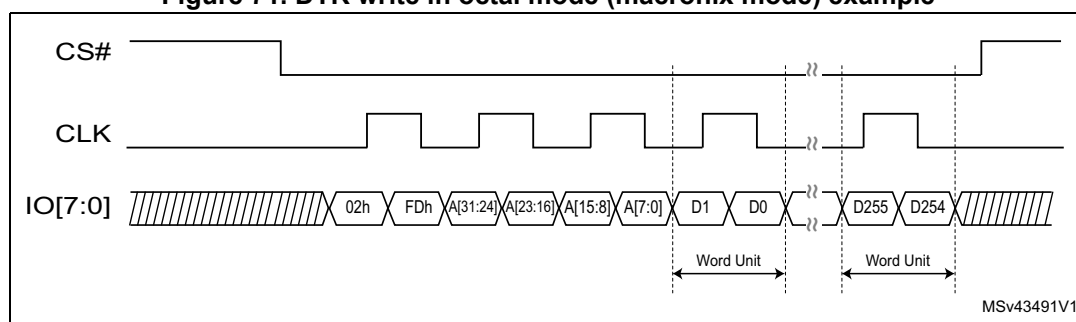
While in memory-mapped mode, the DTR mode for each phases of the write operations are specified in the OCTOSPI\_WCCR register. The DTR mode for each phases of the read operations are specified in the regular register (OCTOSPI\_CCR).

In DTR mode, when the OCTOSPI is driving the IO0/SO, IO1, IO2, IO3, IO4, IO5, IO6 and IO7 signals in the instruction/address/alternate-byte/data phases, a bit is sent or received on each of the falling and rising edges of CLK.

When receiving data in DTR mode, the OCTOSPI assumes that the external device also send the data using both CLK's rising and falling edges. When DDTR=1, the firmware must clear SSHIFT (OCTOSPI\_TCR[30]). Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

In DTR mode, it is recommended to set the delay hold quarter cycle (DHQC) bit of OctoSPI timing configuration register (OCTOSPI\_TCR) to shift the outputs by a quarter of cycle and avoid to hold issues on the memory side.

Figure 71. DTR write in octal mode (macronix mode) example



### Dual-quad mode

When the DQM (OCTOSPI\_CR[6]) bit is 1, the OCTOSPI is in dual-quad mode, where two external Quad-SPI devices (device A and device B) are used in order to send/receive 8-bits (or 16-bits in DTR mode) every cycle, effectively doubling the throughput as well as the capacity.

Each of the devices use the same CLK and nCS signals, but each have a separate IO0, IO1, IO2, and IO3 signals.

The dual-quad mode can be used in conjunction with the single-bit, dual-bit, and quad-bit modes, as well as with either the SDR or the DTR mode.

The device size, as specified in DEVSZ[4:0] (OCTOSPI\_DCR1[20:16]), should reflect the total external device's capacity, which is the double of the size of one individual component.

If address X is even, then the byte that the OCTOSPI gives for address X is the byte at the address X/2 of device A, and the byte which the OCTOSPI gives for address X+1 is the byte at the address X/2 of device B. In other words, the bytes at even addresses are all stored in device A and the bytes at odd addresses are all stored in device B.

When reading the devices' status registers in dual-quad mode, twice as many bytes should be read compared to doing the same read in regular mode. This means that if each device

gives eight valid bits after the instruction for fetching the status register, then the OCTOSPI must be configured with a data length of 2-bytes (16-bits), and the OCTOSPI will receive one byte from each device.

If each device gives a status of 16-bits, then the OCTOSPI must be configured to read four bytes to get the all the status bits of both devices in dual-quad mode. The least-significant byte of the result (in the data register) is the least-significant byte of device A's status register, while the next byte is the least-significant byte of device B's status register. Then, the third byte of the data register is device A's second byte, while the fourth byte is device B's second byte (in the case that the devices have a 16-bit status registers).

An even number of bytes must always be accessed in dual-quad mode. For this reason, bit 0 of the data length field (OCTOSPI\_DLR[0]) is stuck at 1 when DQM=1.

In dual-quad mode, the behavior of device A's interface signals is basically the same as in normal mode. Device B's interface signals have exactly the same waveforms as Device A's during the instruction, address, alternate-byte, and dummy-cycles phases. In other words, each device always receives the same instruction and the same address. Then, during the data phase, the AIOx and the BIOx buses are both transferring data in parallel, but the data that is sent to (or received from) device A is distinct than the one from device B.

### 19.2.3 HyperBus™ mode

While in HyperBus™ mode, the OCTOSPI can communicate with the external device using the HyperBus™ protocol.

The HyperBus™ is using 11 to 12 pins depending on the operating voltage:

- IO[7:0] as bidirectional data bus
- RWDS for read and write data strobe and latency insertion (mapped on DQS pin)
- nCS
- CLK

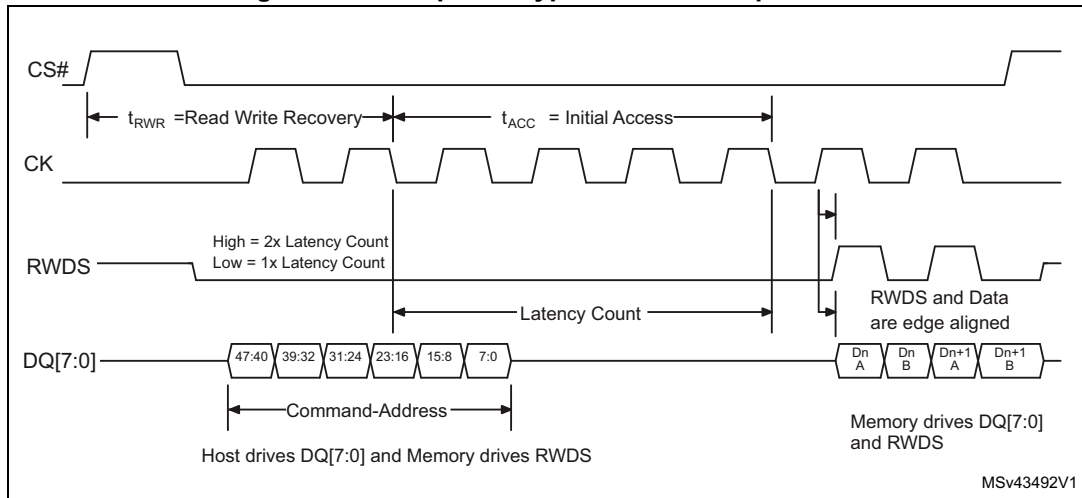
The HyperBus™ does not require any command specification nor any alternate bytes, as a consequence, a separated register set is used to define the timing of the transaction.

The HyperBus™ frame is composed of two phases:

- Command/address phase
- Data phase

The nCS falls before the start of a transaction and rises again after each transaction finishes.

Figure 72. Example of HyperBus™ read operation



The specific HyperBus™ features are configured through the registers in the 0x0200-0x02FC offset range.

### Command/address phase

During this initial phase, the OCTOSPI sends 48 bits over IO[7:0] to specify the operations to be performed with the external device.

Table 110. Command/address phase description

CA bit	Bit Name	Description
47	R/W#	Identify the transaction as a read or a write
46	Address space	Indicates whatever the transaction is accessing the memory or the register space
45	Burst type	Indicates whatever the burst is linear or wrapped
44-16	Row & upper column address	Select the row and the upper column address
15-3	Reserved	-
2-0	Lower column address	Select the starting 16-bit word within the half page

The address space is configured through the memory type (MTYP) field of the OCTOSPI device configuration register (OCTOSPI\_DCR1).

The total size of the device is configured in the device size (DEVSIZE) field of the OCTOSPI device configuration register (OCTOSPI\_DCR1). In case of MCP, the device size is the sum of all the sizes of all the dies of the MCP.

### Read and write operation with initial latency

The HyperBus™ read and write operations need to respect two timings:

- $t_{RWR}$  as minimal read/write recovery time for the device
- $t_{ACC}$  as access time for the device.

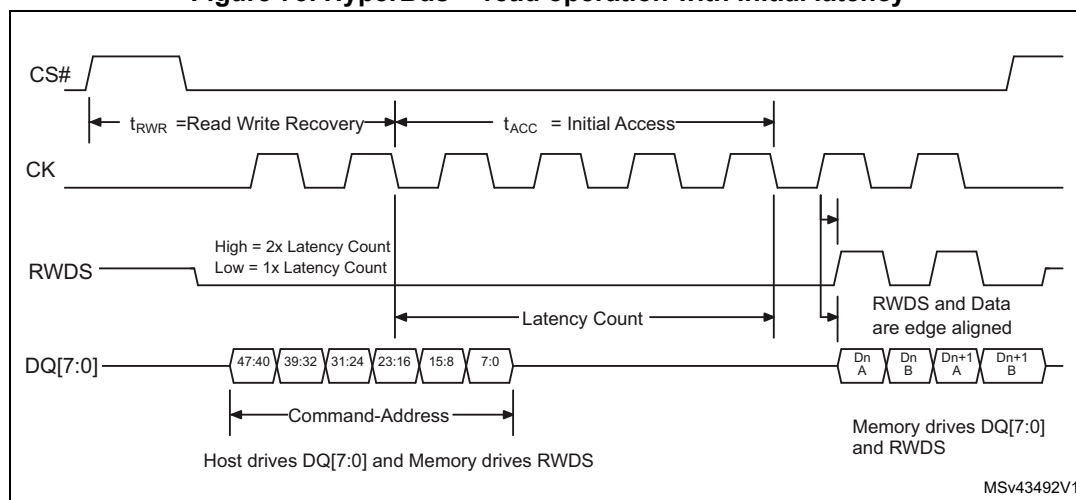
The  $t_{RWR}$  is defined in the TRWR field of the OCTOSPI HyperBus™ latency configuration register (OCTOSPI\_HLCR).

The  $t_{ACC}$  is defined in the TACC field of the OCTOSPI HyperBus™ latency configuration register (OCTOSPI\_HLCR).

During the read operation the RWDS is used in two ways:

- By the device, during the command/address phase, to request an additional latency
- By the device, during the data phase, for data strobing.

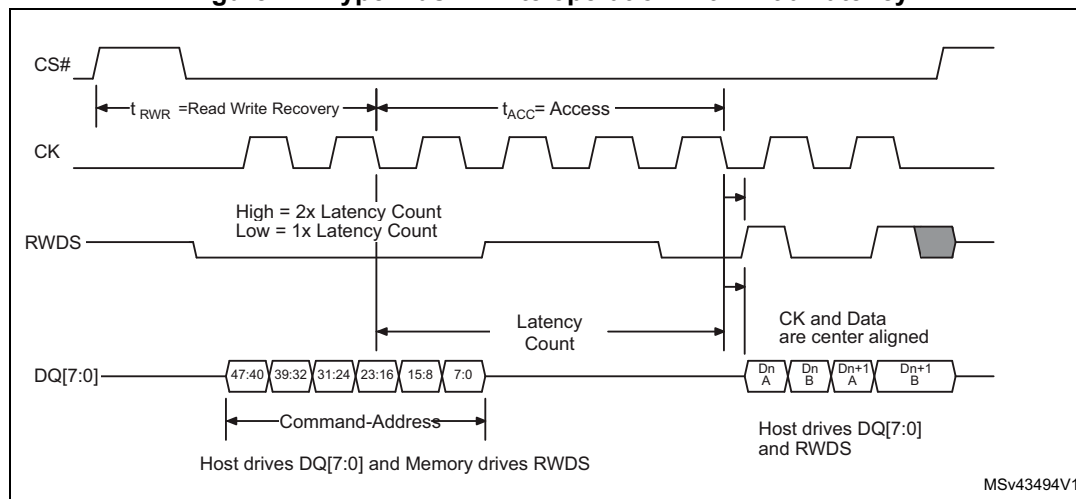
**Figure 73. HyperBus™ read operation with initial latency**



During the write operation the RWDS is used:

- By the device, during the command/address phase, to request an additional latency
- By the OCTOSPI, during the data phase, for write data masking.

**Figure 74. HyperBus™ write operation with initial latency**

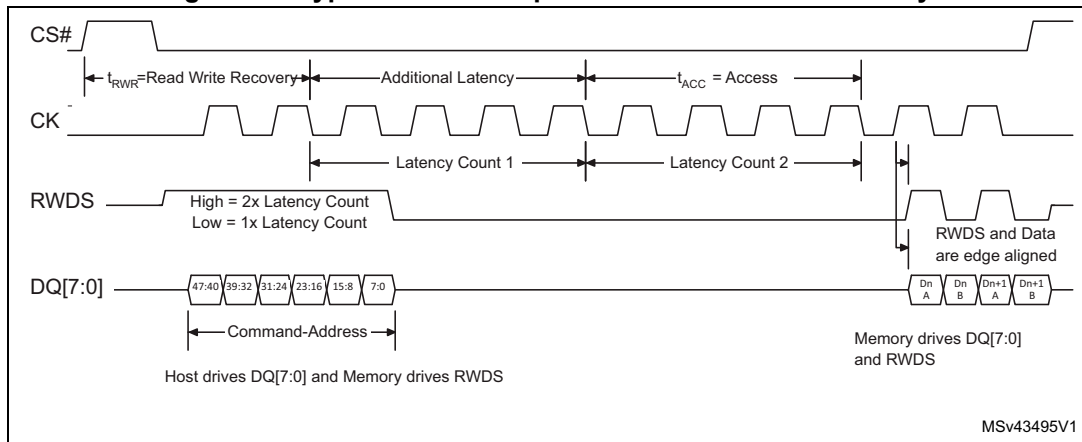


**Read and write operation with additional latency**

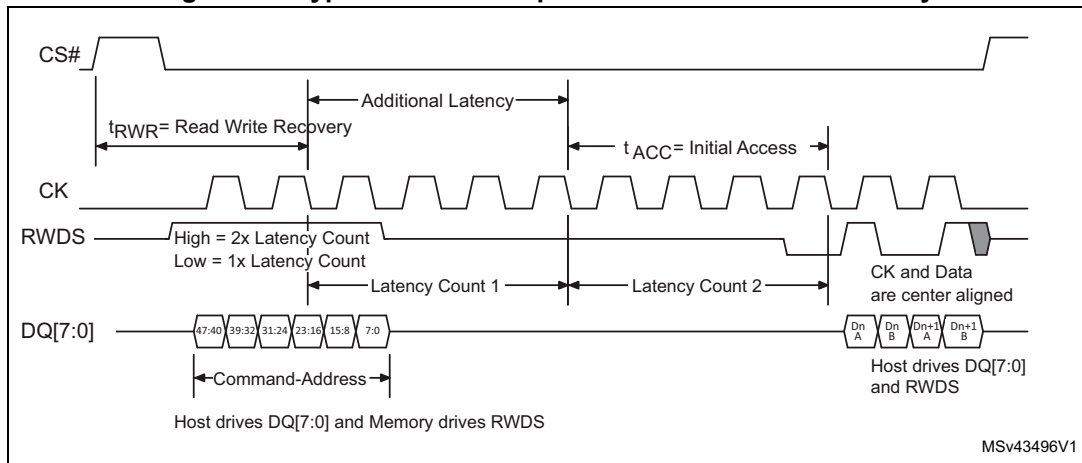
If the device needs an additional latency (during refresh period of a SDRAM for example), it must be tied to one of the RWDS signals during the command/address phase.

An additional  $t_{ACC}$  duration is added by the OCTOSPI to meet the device request.

**Figure 75. HyperBus™ read operation with additional latency**



**Figure 76. HyperBus™ write operation with additional latency**



**Fixed latency mode**

Some devices or some applications may not want to operate with a variable latency time as described above.

It is possible to force the latency to be fixed to  $2 \times t_{ACC}$  by setting the latency mode (LM) bit of the OCTOSPI HyperBus™ latency configuration register (OCTOSPI\_HLCR).

In this fixed latency mode, the state of the RWDS signal is not taken into account by the OCTOSPI and an additional latency is always added leading to a fixed  $2 \times t_{ACC}$  latency time.

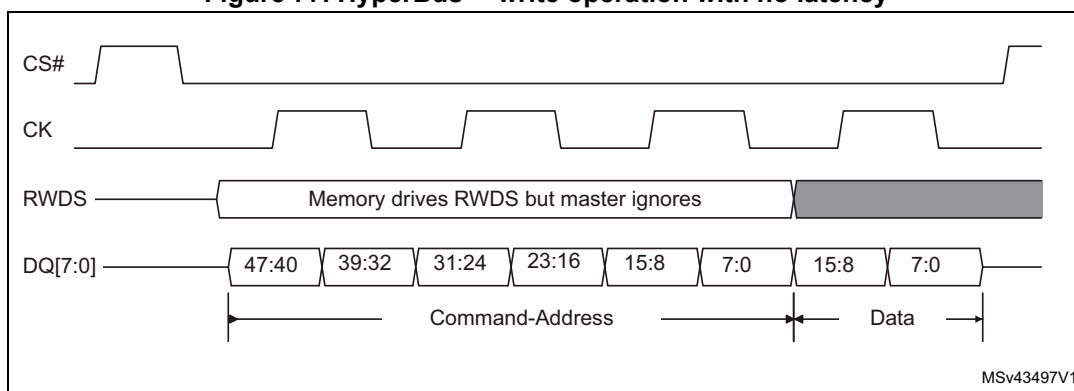
**Write operation with no latency**

Some devices can also require a zero latency for the write operations.

This zero write latency can be forced by setting the write zero latency (WZL) bit of the OCTOSPI HyperBus™ latency configuration register (OCTOSPI\_HLCR).



Figure 77. HyperBus™ write operation with no latency

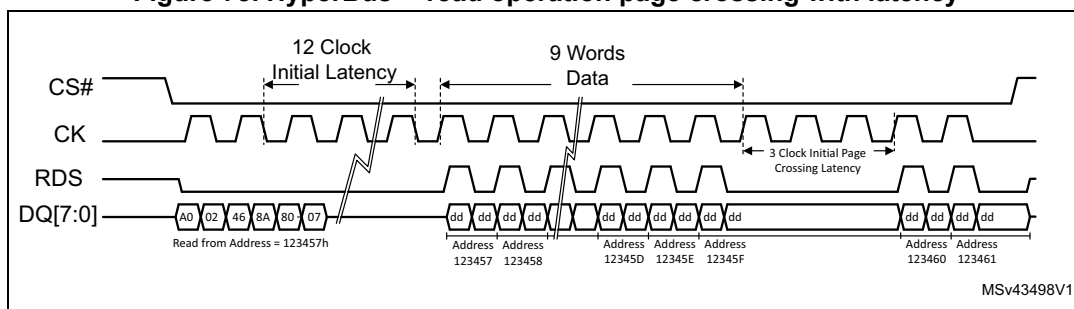


**Latency on page-crossing during the read operations**

An additional latency can be needed by some devices for the read operation when crossing pages.

The initial latency must be respected for any page access, as a consequence, when the first access is close to the page boundary, a latency is automatically added at the page crossing to respect the  $t_{ACC}$  time.

Figure 78. HyperBus™ read operation page crossing with latency



**19.2.4 Common functionality between the regular-command mode and HyperBus™**

The OCTOSPI supports some specific features common to both the regular-command mode and the HyperBus™ such as:

- Wrap support
- CS boundary and regulation

**Wrap support**

The OCTOSPI supports a hybrid wrap as defined by the HyperBus™ protocol. A hybrid wrap is also supported in the regular-command mode.

In hybrid wrap, the transaction can continue after the initial wrap with an incremental access.

The wrap size that is supported by the target memory is configured in the wrap size (WRAPSIZE) field of the OCTOSPI device configuration register 2 (OCTOSPI\_DCR2).

If the target memory is not supporting the hybrid wrap, the WRAPSIZE field must be set to 0.

- 

### **Communication regulation features**

The OCTOSPI transactions can be regulated with the CS boundary feature:

The CS Boundary limits a transaction to a boundary of aligned addresses. The size of the address to be aligned with is configured in the CS boundary (CSBOUND) field of the OCTOSPI device configuration register 3 (OCTOSPI\_DCR3) and it is equal to  $2^{\text{CSBOUND}}$ .

As an example, if CSBOUND = 4, the boundary is set to  $2^4 = 16$  bytes, as a consequence, the nCS is released each time that the LSB address is equal to 0xF and a each time that a new transaction is issued to address the next data.

If CSBOUND is zero, the feature is disabled and a minimum value of "3" is recommended.

The CSBOUND feature can not be used for Flash memory devices in write mode since a command is necessary to program another page of the Flash memory.

## 19.3 OCTOSPI operating modes

The OCTOSPI has three operating modes regardless of the low-level protocol used (it can be either regular or HyperBus™)

- Indirect mode (read or write)
- Status polling mode
- Memory-mapped mode

### 19.3.1 OCTOSPI indirect mode

When in indirect mode, the commands are started by writing to the OCTOSPI registers and the data is transferred by writing or reading the data register, in a similar way to other communication peripherals.

When FMODE=00 (OCTOSPI\_CR[29:28]), the OCTOSPI is in indirect-write mode, where bytes are sent to the external device during the data phase. Data is provided by writing to the data register (OCTOSPI\_DR).

When FMODE=01, the OCTOSPI is in indirect-read mode, where bytes are received from the external device during the data phase. Data is recovered by reading OCTOSPI\_DR.

In indirect mode, when the OCTOSPI is configured in DTR mode over 8 lanes, the given starting address must be even.

The number of bytes to be read/written is specified in the data length register (OCTOSPI\_DLR). If OCTOSPI\_DLR=0xFFFF\_FFFF (all 1's), then the data length is considered undefined and the OCTOSPI simply continues to transfer data until it reaches the end of the external device (as defined by DEVSZ). If no bytes are to be transferred, DMODE (OCTOSPI\_CCR[26:24]) should be set to 00.

If OCTOSPI\_DLR=0xFFFF\_FFFF and DEVSZ=0x1F (its maximum value indicating at 4 GB device), then in this special case the transfers continue indefinitely, stopping only after an abort request or after the OCTOSPI 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 an undefined number of data, the TCF is set when the limit of the external SPI memory is reached according to the device size that is defined in the OCTOSPI\_DCR1 register.

#### Triggering the start of a transfer in regular-command mode

Depending on the OCTOSPI's configuration, there are three different ways to trigger the start of a transfer in indirect mode when using regular mode. In general, it is triggered as

soon as the firmware gives the last information that is necessary for the command, and more specifically, a communication in indirect mode starts at the moment when:

- A write is performed to INSTRUCTION[31:0] (OCTOSPI\_IR), if no address is necessary (when ADMODE=000) and if no data needs to be provided by the firmware (when FMODE=01 or DMODE=000)
- A write is performed to ADDRESS[31:0] (OCTOSPI\_AR), if an address is necessary (when ADMODE!=000) and if no data needs to be provided by the firmware (when FMODE=01 or DMODE=000)
- A write is performed to DATA[31:0] (OCTOSPI\_DR), if an address is necessary (when ADMODE!=000) and if no data needs to be provided by the firmware (when FMODE=00 and DMODE!=000).

A write to the alternate byte register (OCTOSPI\_ABR) never triggers the communication start. If alternate bytes are required, they must have been programmed before.

As soon as a command is started, the BUSY bit (OCTOSPI\_SR[5]) is automatically set.

### Triggering the start of a transfer in HyperBus™ mode

Depending on the OCTOSPI's configuration, there are three different ways to trigger the start of a command in indirect mode. In general, it is triggered as soon as the firmware gives the last information that is necessary for the command, and more specifically, a communication in indirect mode starts at the moment when:

- A write is performed to ADDRESS[31:0] (OCTOSPI\_AR) in indirect-read mode (when FMODE=01)
- A write is performed to DATA[31:0] (OCTOSPI\_DR) in indirect-write mode (when FMODE=00)

As soon as a transfer is started, the BUSY bit (OCTOSPI\_SR[5]) is automatically set.

### FIFO and data management

Data in indirect mode passes through a 32-byte FIFO which is internal to the OCTOSPI. FLEVEL[5:0] (OCTOSPI\_SR[12:8]) 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 in the OCTOSPI\_DR. A 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 at DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

The byte/halfword accesses to the OCTOSPI\_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 after which point the FIFO threshold flag, FTF, gets set. In indirect-read mode, the FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. The FTF is also set if there is any data left in the FIFO after the last byte is read from the external device, regardless of the FTHRES setting. In indirect-write mode, the FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE=1, there is an interrupt when the FTF is set. If DMAEN=1, a DMA transfer is initiated when the FTF is set. The FTF is cleared by HW as soon as the threshold condition is no longer true (after enough data has been transferred by the CPU or DMA).

In indirect-read mode when the FIFO becomes full, the OCTOSPI temporarily stops reading bytes from the external device to avoid an overrun. Note that the reading of the external device does not restart until 4 bytes become vacant in the FIFO (when  $FLEVEL \leq 28$ ). Thus, when  $FTHRES \geq 29$ , the application must be sure to read enough bytes to assure that the OCTOSPI starts retrieving data from the external device again. Otherwise, the FTF flag stays at '0' as long as  $FLEVEL < FTHRES$ .

### 19.3.2 OCTOSPI status flag polling mode

In automatic-polling mode, the OCTOSPI 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.

The accesses to the device begins in the same manner as in indirect-read mode. BUSY goes high at this point and stays high even between the periodic accesses.

The contents of MASK[31:0] (OCTOSPI\_PSMAR) are used to mask the data from the external device in automatic-polling mode. If the 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] (OCTOSPI\_PSMAR) then there is a match for bit n.

If the polling-match mode bit (PMM, OCTOSPI\_CR[23]) is 0, then “AND” match mode is activated. This means that only when there is a match on all of the unmasked bits, then the status match flag (SMF) is set.

If PMM=1, then “OR” match mode is activated, which means that the SMF gets set if there is a match on any of the unmasked bits.

An interrupt is called when the SMF is set if SMIE=1.

If the automatic-polling mode stop (APMS) bit is set, then 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 until the OCTOSPI is disabled (EN=0).

The data register (OCTOSPI\_DR) contains the latest received status bytes (the FIFO is deactivated). The contents of the data register are 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 the FTF is cleared as soon as the data is read.

In automatic-polling mode, variable latency is not supported. As a consequence, the memory must be configured in fixed latency.

### 19.3.3 OCTOSPI memory-mapped mode

When configured in memory-mapped mode, the external SPI device is seen as an internal memory. Note that no more than 256 MB can be addressed even if the external device capacity is larger.

If an access is made to an address outside of the range defined by DEVSIZE but still within the 256 MB range, then an AHB error is given. The effect of this error depends on the AHB master that attempted the access:

- If it is the Cortex CPU, a hard fault interrupt is generated
- If it is a DMA, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

Byte, halfword, and word access types are all supported.

A support for execute in place (XIP) operation is implemented, where the OCTOSPI continues to load the bytes to the addresses following the most recent access. If subsequent accesses are continuous to the bytes that follow, then these operations ends up quickly since their results were pre-fetched.

By default, the OCTOSPI never stops its prefetch operation, it either keeps the previous read operation active with the nCS maintained low or it relaunches a new transfer, even if no access to the external device occurs for a long time.

Since external devices tend to consume more when the nCS is held low, the application might want to activate the timeout counter (TCEN=1, OCTOSPI\_CR[3]) so that the nCS is released after a period of TIMEOUT[15:0] (OCTOSPI\_LPTR), when x cycles have elapsed without an access since the clock is inactive.

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.

## 19.4 OCTOSPI configuration

The OCTOSPI configuration is done in three steps:

1. OCTOSPI system configuration
2. OCTOSPI device configuration
3. OCTOSPI mode configuration

### 19.4.1 OCTOSPI system configuration

The OCTOSPI is configured using the OCTOSPI\_CR. The user must program:

- Functional mode (FMODE field)
- Polling mode behavior if needed (PMM and APMS bits)
- FIFO level for either DMA request generation or interrupt generation (FTHRES field)
- DMA usage (DMAEN bit)
- Timeout counter usage (TCEN bit)
- Dual-quad mode (DQM bit) if needed (only for Quad-SPI configuration)

In case of an interrupt usage, their respective enable bit can also be set during this phase.

If the timeout counter is used, the timeout value is programmed in the OCTOSPI\_LPTR register.

The DMA channel must not be enabled during the OCTOSPI configuration, it must be enabled only once the operation is fully configured to avoid any unexpected request generation.

### 19.4.2 OCTOSPI device configuration

The parameters related to the external device targeted are configured through the OCTOSPI\_DCR1 and the OCTOSPI\_DCR2 registers. The user must program:

- Memory type (MTYP field)
- Device size (DEVSIZE field)
- Chip-select minimum high time (CSHT field)
- Clock mode (FRCK and CKMODE bit)
- Supported wrap size (WRAPSIZE field)
- Device frequency (PRESCALER field)
- The MTYP[2:0] field defines the type of memory to be used for 8-line modes:
- Micron mode with D0 / D1 ordering in 8-data-bit mode (DMODE = 100)
- Macronix mode with D1 / D0 ordering in 8-data-bit mode (DMODE = 100)
- HyperBus™ memory mode. Protocol follows the Spansion's HyperBus™ specification, and an 8-data-bit DDR mode must be selected.
- HyperBus™ register mode. Addressing register space, the memory-mapped accesses in this mode must be non-cacheable, or the indirect read/write modes must be used.

The DEVSIZE[4:0] field defines the size of external memory using the following formula:

$$\text{Number of bytes in the device} = 2^{[\text{DEVSIZE}+1]}$$

DEVSIZE+1 is effectively the number of address bits required to address the external device. The external device capacity can go up to 4 GB (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 MB.

If DQM=1, then DEVSZ indicates the total capacity of the two devices together.

When the OCTOSPI executes two commands, one immediately after the other, it raises the chip-select signal (nCS) high between the two commands for only one CLK cycle by default.

If the external device requires more time between commands, the chip-select high time (CSHT) field can be used to specify the minimum number of CLK cycles (up to 8) for which the nCS must remain high.

The clock mode (CKMODE) bit indicates the level that the CLK takes between commands (when nCS=1).

In HyperBus™ mode, the device timing ( $t_{ACC}$  and  $t_{RWR}$ ) and the latency mode must be configured in the OCTOSPI HyperBus™ latency configuration register (OCTOSPI\_HLCR).

### 19.4.3 OCTOSPI regular-command mode configuration

#### Indirect mode configuration

When FMODE is programmed to 00, the indirect-write mode is selected and the data can be sent to the external device. When FMODE=01, the indirect-read mode is selected where data can be read from the external device.

When the OCTOSPI 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 the OCTOSPI\_DLR
2. Specify the frame timing in the OCTOSPI\_TCR
3. Specify the frame format in the OCTOSPI\_CCR
4. Specify the instruction in the OCTOSPI\_IR
5. Specify the optional alternate byte to be sent right after the address phase in the OCTOSPI\_ABR
6. Specify the targeted address in the OCTOSPI\_AR
7. Unenable the DMA channel if needed
8. Read/write the data from/to the FIFO through the OCTOSPI\_DR (if no DMA usage)

If neither the address register (OCTOSPI\_AR) nor the data register (OCTOSPI\_DR) need to be updated for a particular command, then the command sequence starts as soon as the OCTOSPI\_CCR is written. This is the case when both ADMODE and DMODE are 000, or if just ADMODE=000 when in indirect-read mode (FMODE=01).

When an address is required (ADMODE is not 000) and the data register does not need to be written (when FMODE=01 or DMODE=000), the command sequence starts as soon as the address is updated with a write to OCTOSPI\_AR.

In case of data transmission (FMODE=00 and DMODE!=000), the communication start is triggered by a write in the FIFO through the OCTOSPI\_DR.

#### Status flag polling mode configuration

The status flag polling mode is enabled by setting the FMODE field (OCTOSPI\_CR[29:28]) to 10. In this mode, the programmed frame is sent and the data is retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in the OCTOSPI\_DLR, it will be ignored and only 4 bytes will be read.

The periodicity is specified in the OCTOSPI\_PIR register.



Once the status data has been retrieved, it can internally be processed to:

- Set the SMF (status match flag) and generate an interrupt if enabled
- Stop automatically the periodic retrieving of the status bytes.

The received value can be masked with the value stored in the OCTOSPI\_PSMKR and in ORed or ANDed with the value stored in the OCTOSPI\_PSMAR.

In case of a match, the SMF is set and an interrupt is generated if enabled, and, the OCTOSPI can be automatically stopped if the AMPS bit is set. In any case, the latest retrieved value is available in the OCTOSPI\_DR.

When the OCTOSPI is used in auto-polling mode, the frames are constructed in the following way:

1. Specify the input mask in the OCTOSPI\_PSMKR
2. Specify the comparison value in the OCTOSPI\_PSMAR
3. Specify the read period in the OCTOSPI\_PIR
4. Specify a number of data bytes to read in the OCTOSPI\_DLR
5. Specify the frame timing in the OCTOSPI\_TCR
6. Specify the frame format in the OCTOSPI\_CCR
7. Specify the instruction in the OCTOSPI\_IR
8. Specify the optional alternate byte to be sent right after the address phase in the OCTOSPI\_ABR
9. Specify the optional targeted address in the OCTOSPI\_AR.

If the address register (OCTOSPI\_AR) does not need to be updated for a particular command, then the command sequence starts as soon as the OCTOSPI\_CCR is written. This is the case when ADMODE=000.

When an address is required (ADMODE is not 000), the command sequence starts as soon as the address is updated with a write to the OCTOSPI\_AR.

### Memory-mapped mode configuration

In memory-mapped mode, the external device is seen as an internal memory but with some latency during accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to do Flash programming using memory-mapped write as internal flags for erase or programming status have to be polled.

Memory-mapped write operation are not supported on memories without write strobe or in dual-quad mode.

Memory-mapped mode is entered by setting the FMODE to 11 in the OCTOSPI\_CR register.

The programmed instruction and frame are sent when an AHB master is accessing the memory mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to the OCTOSPI\_DR in this mode returns zero.

The data length register (OCTOSPI\_DLR) has no meaning in memory-mapped mode.

When the OCTOSPI is used in memory-mapped mode, the frames are constructed in the following way:

1. Specify the frame timing in the OCTOSPI\_TCR for read operation
2. Specify the frame format in the OCTOSPI\_CCR for read operation
3. Specify the instruction in the OCTOSPI\_IR
4. Specify the optional alternate byte to be sent right after the address phase in the OCTOSPI\_ABR for read operation
5. Specify the frame timing in the OCTOSPI\_WTCR for write operation
6. Specify the frame format in the OCTOSPI\_WCCR for write operation
7. Specify the instruction in the OCTOSPI\_WIR
8. Specify the optional alternate byte to be sent right after the address phase in the OCTOSPI\_WABR for read operation

All the configuration operations must be completed before the first access to the memory area. On the first access, the OCTOSPI becomes busy, and no further configuration is allowed.

### **OCTOSPI delayed data sampling**

By default, when no DQS, is used, the OCTOSPI samples the data driven by the external device one half of a CLK cycle after the external device drives the signal.

In case of any external signal delays, it may be useful to sample the data later. Using the SSHIFT (OCTOSPI\_TCR[30]) bit, the sampling of the data can be shifted by half of a CLK cycles.

The firmware must clear the SSHIFT when the data phase is configured in DTR mode (DDTR=1).

### **Sending the instruction only once**

A Flash memory (like Winbond) might provide a mode where an instruction must be sent only with the first command sequence, while subsequent commands start directly with the address. The user can take advantage of this type of features using the SIOO bit (OCTOSPI\_CCR[31]).

The SIOO is valid for indirect mode only. If the SIOO bit is set, the instruction is sent only for the first command following a write to OCTOSPI\_CCR.

Subsequent command sequences skip the instruction phase, until there is a write to OCTOSPI\_CCR. The SIOO has no effect when IMODE=00 (no instruction).

## **19.4.4 OCTOSPI HyperBus™ mode configuration**

### **Indirect mode configuration**

When the FMODE is programmed to 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE=01, the indirect-read mode is selected where data can be read from the external device.

When the OCTOSPI 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 the OCTOSPI\_DLR
2. Specify the targeted address in the OCTOSPI\_AR
3. Unenable the DMA channel if needed
4. Read/write the data from/to the FIFO through the OCTOSPI\_DR (if no DMA usage).

In indirect-read mode (when FMODE=01), the command sequence starts as soon as the address is updated with a write to OCTOSPI\_AR.

In indirect-write mode (when FMODE=00), the communication start is triggered by a write in the FIFO through OCTOSPI\_DR.

### Memory-mapped mode configuration

In memory-mapped mode, the external device is seen as an internal memory but with some latency during the accesses. Read and write operations are allowed to the external device in this mode.

The memory-mapped mode is entered by setting the FMODE to 11 in the OCTOSPI\_CR register. The programmed instruction and frame is sent when an AHB master is accessing the memory mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to the OCTOSPI\_DR in this mode returns zero.

The data length register (OCTOSPI\_DLR) has no meaning in memory-mapped mode.

All the configuration operation must be completed prior to the first access to the memory area. On the first access, the OCTOSPI becomes busy, and no configuration is allowed.

## 19.4.5 OCTOSPI error management

A error can be generated in the following cases:

- In indirect mode or status flag polling mode when a wrong address has been programmed in the OCTOSPI\_AR (according to the device size defined by DEVSZ[4:0] in the OCTOSPI\_DCR1). This sets the TEF and an interrupt is generated if enabled.
- In indirect mode, if the address plus the data length exceed the device size, the TEF is set as soon as the access is triggered.
- In memory-mapped mode when an out of range access is done by an AHB master, it generates an AHB error as a response to the faulty AHB request.
- An access to the memory-mapped area when the memory-mapped mode is disabled, it generates an AHB error as a response to the faulty AHB request.

## 19.4.6 OCTOSPI busy bit and abort functionality

Once the OCTOSPI starts an operation with the external device, the BUSY bit is automatically set in the OCTOSPI\_SR.

In indirect mode, the BUSY bit is reset once the OCTOSPI has completed the requested command sequence and the FIFO is empty.

In automatic-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 the OCTOSPI\_CR. Once the abort is completed, the BUSY bit and the ABORT bit are automatically reset, and the FIFO is flushed.

Before setting the ABORT bit, the software must ensure that all the current transactions are finished using the synchronization barriers.

Note that some devices might misbehave if a write operation to a status registers is aborted.

### 19.4.7 OCTOSPI reconfiguration or deactivation

Prior to any OCTOSPI reconfiguration, the software must ensure that all the transactions are completed:

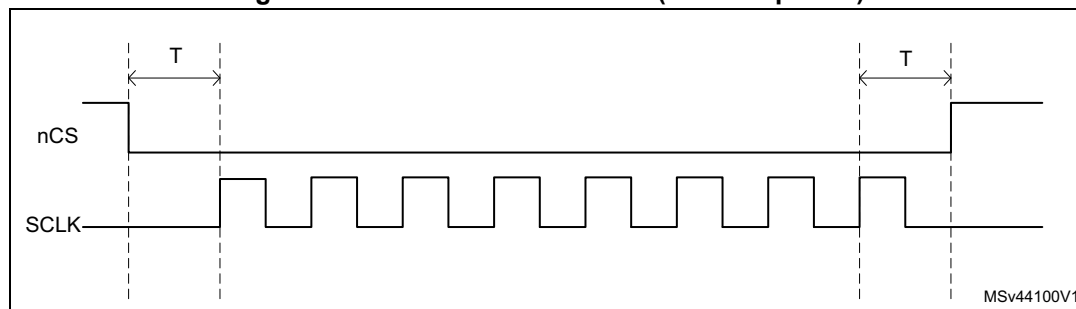
- After a memory-mapped write, the software shall perform a dummy read followed by a synchronization barrier then an ABORT.
- After a memory-mapped read, the software shall do a synchronization barrier then an abort.

### 19.4.8 nCS behavior

By default, the nCS is high, deselecting the external device. The nCS falls before an operation begins and rises as soon as it finishes.

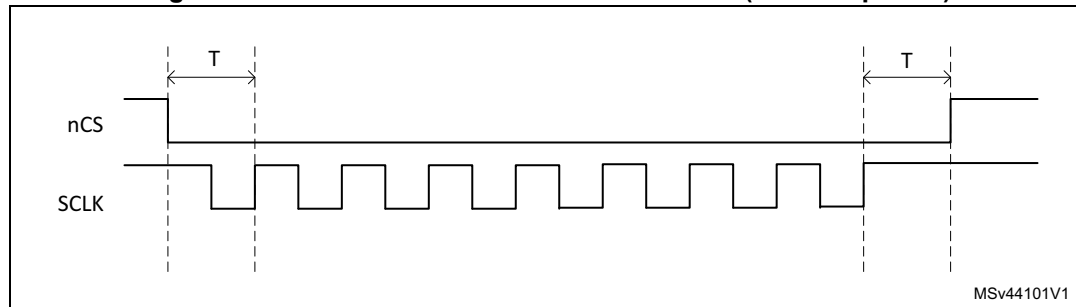
When the CKMODE=0 (“mode0”, where CLK stays low when no operation is in progress), the nCS falls one CLK cycle before an operation’s first rising CLK edge, and the nCS rises one CLK cycle after the operation’s final rising CLK edge, as shown in [Figure 79](#).

Figure 79. nCS when CKMODE=0 (T = CLK period)



When the CKMODE=1 (“mode3”, where CLK goes high when no operation is in progress) and when in SDR mode, the nCS falls one CLK cycle before an operation’s first rising CLK edge, and the nCS rises one CLK cycle after the operation’s final rising CLK edge, as shown in [Figure 80](#).

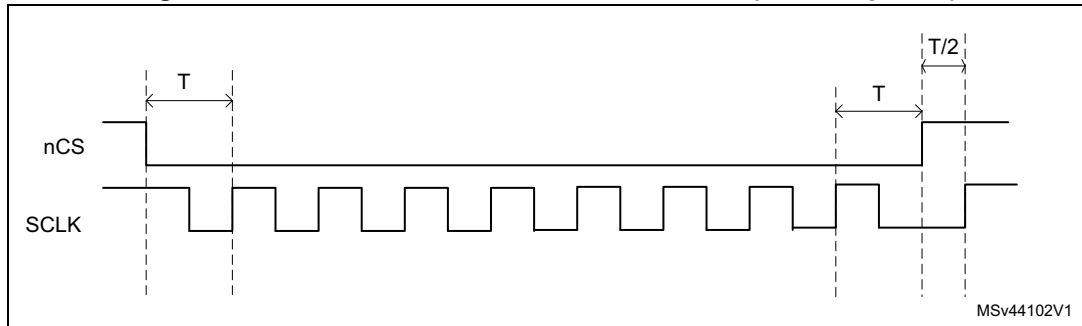
Figure 80. nCS when CKMODE=1 in SDR mode (T = CLK period)



When the CKMODE=1 (“mode3”) and the DDTR=1 (data DTR mode), the nCS falls one CLK cycle before an operation’s first rising CLK edge, and the nCS rises one CLK cycle after the operation’s final active rising CLK edge, as shown in [Figure 81](#). Because the DDR

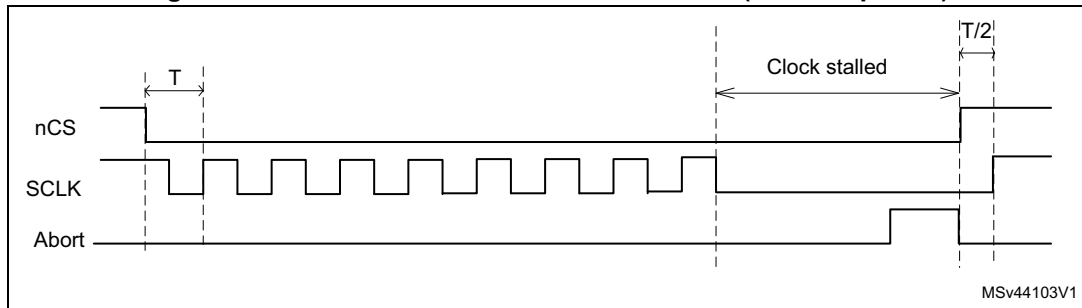
operations must finish with a falling edge, the CLK is low when the nCS rises, and the CLK rises back up one half of a CLK cycle afterwards.

**Figure 81. nCS when CKMODE=1 in DDTR mode (T = CLK period)**



When the FIFO stays full during a read operation or if the FIFO stays empty during a write operation, the operation stalls and the CLK stays low until the firmware services the FIFO. If an abort occurs when an operation is stalled, the nCS rises just after the abort is requested and then the CLK rises one half of a CLK cycle later, as shown in [Figure 82](#).

**Figure 82. nCS when CKMODE=1 with an abort (T = CLK period)**



When not in dual-quad mode (DQM=0), only device A is accessed and thus the BnCS stays high. In dual-quad mode, the BnCS behaves exactly the same as the AnCS. Thus, if there is a device B and if the application always stays in dual-quad mode, then the device B may use the AnCS and the pin outputting BnCS can be used for other functions.

## 19.5 OCTOSPI 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 to provide more flexibility.

**Table 111. OCTOSPI 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

## 19.6 OCTOSPI registers

### 19.6.1 OCTOSPI control register (OCTOSPI\_CR)

Address offset: 0x0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.		FMODE		Res.				PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE
		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.			FTHRES[4:0]				FSEL	DQM	Res.		TCEN	DMAEN	ABORT	EN	
			rw				rw	rw			rw	w1s	rw	w1s	

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

Bits 29:28 **FMODE[1:0]**: Functional mode

This field defines the OCTOSPI's functional mode of operation

00: Indirect-write mode

01: Indirect-read mode

10: Automatic-polling mode

11: Memory-mapped mode

If DMAEN=1 already, then the DMA controller for the corresponding channel must be disabled before changing the FMODE value

This field can be written only when BUSY=0

Bits 27:24 Reserved, must be kept at reset value

Bit 23 **PMM**: Polling match mode

This bit indicates which method should be used to determine a “match” during the automatic-polling mode.

0: AND match mode, SMF is set if all the unmasked bits received from the device match the corresponding bits in the match register

1: OR match mode, SMF is set if any of the unmasked bits received from the device matches its corresponding bit in the match register

This bit can be modified only when BUSY=0

Bit 22 **APMS**: Automatic poll mode stop

This bit determines if the automatic polling is stopped after a match

0: Automatic-polling mode is stopped only by abort or by disabling the OCTOSPI

1: Automatic-polling mode stops as soon as there is a match

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 disable

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  
Defines, in indirect mode, the threshold number of bytes in the FIFO which will cause the FIFO threshold flag (FTF, OCTOSPI\_SR[2]) to be set  
In indirect-write mode (FMODE=00):  
0: FTF is set if there are 1 or more free bytes available to be written to in the FIFO  
1: FTF is set if there are 2 or more free bytes available to be written to in the FIFO  
...  
31: FTF is set if there are 32 free bytes available to be written to in the FIFO  
  
In indirect-read mode (FMODE=01):  
0: FTF is set if there are 1 or more valid bytes can be read from the FIFO  
1: FTF is set if there are 2 or more valid bytes can be read from the FIFO  
...  
31: FTF is set if there are 32 valid bytes 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
- Bit 7 **FSEL**: Flash select  
This bit selects the Flash memory to be addressed in single/dual/quad mode in single-Flash mode (when DFM = 0)  
0: FLASH 1 selected (data exchanged over IO[3:0])  
1: FLASH 2 selected (data exchanged over IO[3:0])  
This bit is ignored when DFM = 1 or when octal mode is selected  
This bit can be modified only when BUSY=0
- Bit 6 **DQM**: Dual-quad mode  
This bit activates the dual-quad mode, where two external devices are used simultaneously to double the throughput and the capacity  
0: Dual-quad mode disabled  
1: Dual-quad mode enabled  
This bit can be modified only when BUSY=0.
- Bits 5:4 Reserved, must be kept at reset value



- Bit 3 **TCEN**: Timeout counter enable  
 This bit is valid only when the memory-mapped mode (FMODE=11) is selected. Enables the timeout counter  
 0: Timeout counter is disabled, and thus the chip-select (nCS) remains active indefinitely after an access in memory-mapped mode  
 1: Timeout counter is enabled, and thus the chip-select is released in the memory-mapped mode after TIMEOUT[15:0] cycles of external device inactivity  
 This bit can be modified only when BUSY=0
  
- Bit 2 **DMAEN**: DMA enable  
 In indirect mode, DMA can be used to input or output data via the OCTOSPI\_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 on-going command sequence. It is automatically reset once the abort is completed  
 This bit stops the current transfer  
 In polling mode or memory-mapped mode, this bit also resets the APM bit or the DM bit  
 0: No abort requested  
 1: Abort requested
  
- Bit 0 **EN**: Enable  
 Enables the OCTOSPI  
 0: OCTOSPI is disabled  
 1: OCTOSPI is enabled

### 19.6.2 OCTOSPI device configuration register 1 (OCTOSPI\_DCR1)

Address offset: 0x0008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.					MTYP			Res.			DEVSIZE				
					rw						rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.					CSHT			Res.					FRCK	CK-MODE	
					rw								rw	rw	

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

Bits 26:24 **MTYP**: Memory type

This bit indicates the type of memory to be supported

000: Micron mode, D0/D1 ordering in DTR 8-data-bit mode. Regular SPI protocol in octal/quad/dual/single modes

001: Macronix mode, D1/D0 ordering in DTR 8-data-bit mode. Regular SPI protocol in octal/quad/dual/single modes

010-011: Reserved

010: APMemory mode with reversed DQS polarity

011: Macronix RAM mode, D1/D0 ordering in DTR 8-data-bit mode. Regular SPI protocol in octal/quad/dual/single modes with dedicated address mapping

100: HyperBus™ memory mode, the protocol follows the Spansion's HyperBus™ specification. 8-data-bit DDR mode must be selected

101: HyperBus™ register mode, addressing register space. The memory-mapped accesses in this mode must be non-cacheable, or indirect read/write modes must be used

110-111: Reserved, must be kept at reset value

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

Bits 20:16 **DEVSZ[4:0]**: Device size

This field defines the size of the external device using the following formula:

Number of bytes in device =  $2^{[DEVSZ+1]}$

DEVSZ+1 is effectively the number of address bits required to address the external device

The device capacity can be up to 4 GB (addressed using 32-bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 MB

In regular-command mode, if DQM=1, DEVSZ indicates the total capacity of the two devices together

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 where the chip-select (nCS) must remain high between commands issued to the external device

0: nCS stays high for at least 1 cycle between external device commands

1: nCS stays high for at least 2 cycles between external device commands

...

7: nCS stays high for at least 8 cycles between external device commands

Bits 7:2 Reserved, must be kept at reset value

Bit 1 **FRCK**: Free running clock

This bit configures the free running clock

0: CLK is not free running

1: CLK is free running (always provided)

Bit 0 **CKMODE**: Mode 0 / mode 3

This bit indicates the level that the 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

### 19.6.3 OCTOSPI device configuration register 2 (OCTOSPI\_DCR2)

Address offset: 0x000C

Reset value: 0x0000 0000



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

Bits 18:16 **WRAPSIZE**: Wrap size

This field indicates the wrap size to which the memory is configured.

000: Wrapped reads are not supported by the memory

001: Reserved, must be kept at reset value

010: External memory supports wrap size of 16 bytes (important)

011: External memory supports wrap size of 32 bytes (most likely)

100: External memory supports wrap size of 64 bytes (L1 MPU)

101: External memory supports wrap size of 128 bytes

110-111: Reserved, must be kept at reset value

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

Bits 7:0 **PRESCALER[7:0]**: Clock prescaler

This field defines the scaler factor for generating the CLK based on the Kernel clock (value+1)

0:  $F_{CLK} = F_{KERNEL}$ , AHB clock used directly as OCTOSPI's CLK (prescaler bypassed)

1:  $F_{CLK} = F_{KERNEL}/2$

2:  $F_{CLK} = F_{KERNEL}/3$

...

255:  $F_{CLK} = F_{KERNEL}/256$

For odd clock division factors, the CLK's duty cycle is not 50%. The clock signal remains high one cycle longer than it stays low

This field can be modified only when BUSY=0

### 19.6.4 OCTOSPI device configuration register 3 (OCTOSPI\_DCR3)

Address offset: 0x0010

Reset value: 0x0000 0000



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

Bits 20:16 **CSBOUND[4:0]**: CS boundary

This bits enable the transaction boundary feature. When active, a minimum value of 3 is recommended

The nCS is released on each boundary of  $2^{CSBOUND}$  bytes

0: CS boundary disabled

others: CS boundary set to  $2^{CSBOUND}$  bytes

Bits 15:0 Reserved, must be kept at reset value

### 19.6.5 OCTOSPI status register (OCTOSPI\_SR)

Address offset: 0x0020

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.		FLEVEL[5:0]					Res.		BUSY	TOF	SMF	FTF	TCF	TEF	
		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 that 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 ongoing

This bit clears automatically when the operation with the external device 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-polling mode when the unmasked received data matches the corresponding bits in the match register (OCTOSPI\_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 has been reached, or if there is any data left in the FIFO after the reads from the external device are complete

It is cleared automatically as soon as the threshold condition is no longer true

In automatic-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 has been transferred or in any mode when the transfer has been 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

### 19.6.6 OCTOSPI flag clear register (OCTOSPI\_FCR)

Address offset: 0x0024

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
												w1o	w1o		w1o	w1o

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

Bit 4 **CTOF**: Clear timeout flag

Writing 1 clears the TOF flag in the OCTOSPI\_SR register

Bit 3 **CSMF**: Clear status match flag

Writing 1 clears the SMF flag in the OCTOSPI\_SR register

Bit 2 Reserved, must be kept at reset value

Bit 1 **CTCF**: Clear transfer complete flag

Writing 1 clears the TCF flag in the OCTOSPI\_SR register

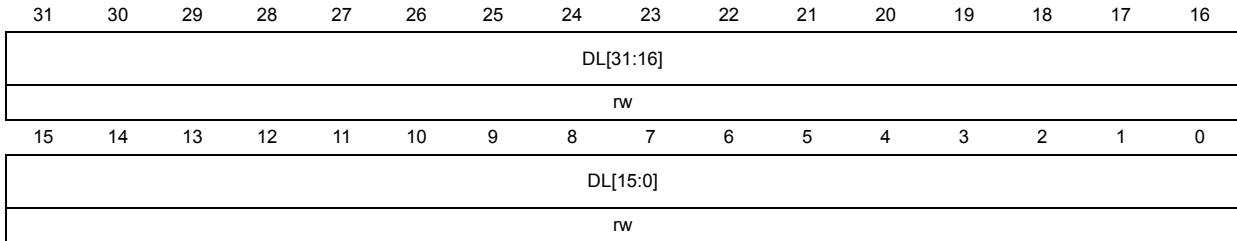
Bit 0 **CTEF**: Clear transfer error flag

Writing 1 clears the TEF flag in the OCTOSPI\_SR register

### 19.6.7 OCTOSPI data length register (OCTOSPI\_DLR)

Address offset: 0x0040

Reset value: 0x0000 0000



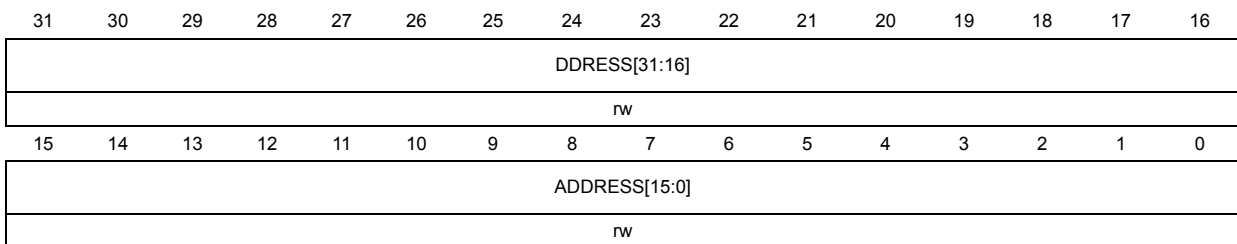
Bits 31:0 **DL[31: 0]**: Data length

Number of data to be retrieved (value+1) in indirect and status-polling modes. A value no greater than 3 (indicating 4 bytes) should be used for status-polling mode  
 All 1s in indirect mode means undefined length, where OCTOSPI continues until the end of the memory, as defined by DEVSZIE  
 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 the external device, (as defined by DEVSZIE) are to be transferred. Continue reading indefinitely if DEVSZIE=0x1F  
 DL[0] is stuck at '1' in dual-quad mode (DQM=1) even when '0' is written to this bit, thus assuring that each access transfers an even number of bytes  
 This field has no effect when in memory-mapped mode (FMODE=10)  
 This field can be written only when BUSY=0

### 19.6.8 OCTOSPI address register (OCTOSPI\_AR)

Address offset: 0x0048

Reset value: 0x0000 0000



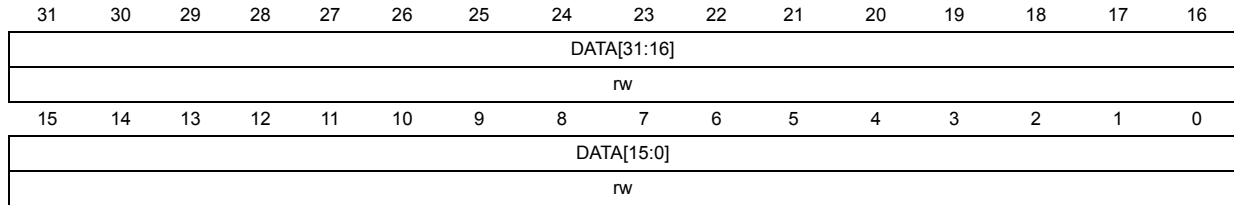
Bits 31:0 **ADDRESS[31 0]**: Address

Address to be sent to the external device  
 In Hyperflash™ mode, this field must be even as this protocol is 16-bit word oriented  
 Writes to this field are ignored when BUSY=0 or when FMODE=11 (memory-mapped mode)

### 19.6.9 OCTOSPI data register (OCTOSPI\_DR)

Address offset: 0x0050

Reset value: 0x0000 0000



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 external device 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 external device. 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-polling mode, this register contains the last data read from the external device (without masking)

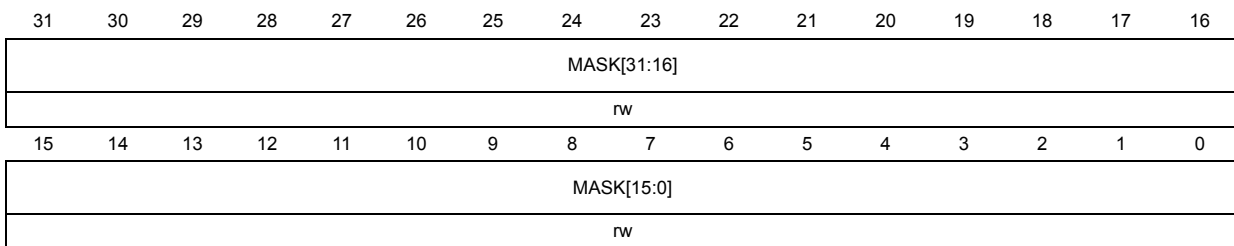
Word, halfword, and byte accesses to this register are supported. In indirect-write mode, a byte write adds 1 byte to the FIFO, a halfword write 2, and a word write 4

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]

### 19.6.10 OCTOSPI polling status mask register (OCTOSPI\_PSMKR)

Address offset: 0x0080

Reset value: 0x0000 0000



Bits 31:0 **MASK[31: 0]**: Status mask

Mask to be applied to the status bytes received in polling mode

For bit n:

0: Bit n of the data received in automatic-polling mode is masked and its value is not considered in the matching logic

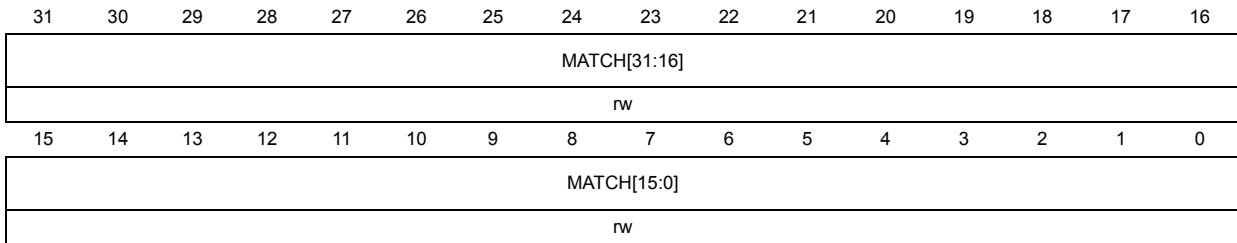
1: Bit n of the data received in automatic-polling mode is unmasked and its value is considered in the matching logic

This field can be written only when BUSY=0

### 19.6.11 OCTOSPI polling status match register (OCTOSPI\_PSMAR)

Address offset: 0x0088

Reset value: 0x0000 0000



Bits 31:0 **MATCH[31: 0]**: Status match

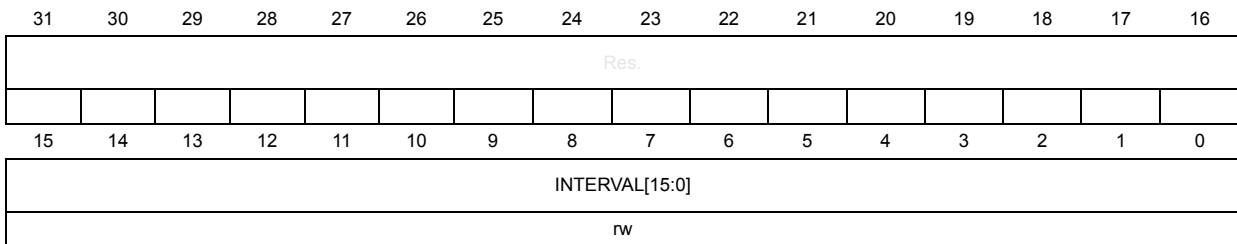
Value to be compared with the masked status register to get a match

This field can be written only when BUSY=0

### 19.6.12 OCTOSPI polling interval register (OCTOSPI\_PIR)

Address offset: 0x0090

Reset value: 0x0000 0000



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

Bits 15:0 **INTERVAL[15: 0]**: Polling interval

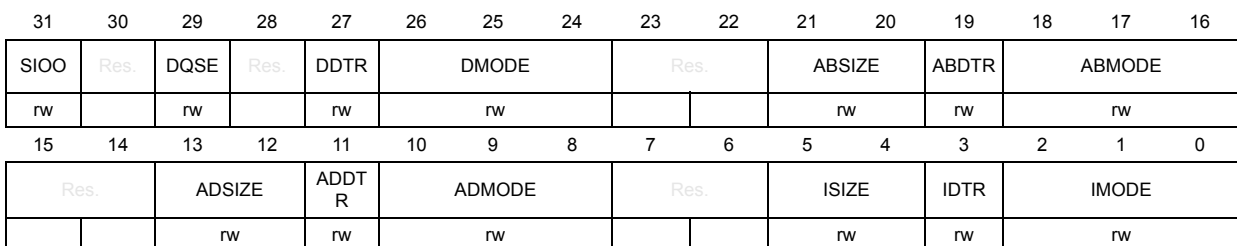
Number of CLK cycle between a read during the automatic polling phases

This field can be written only when BUSY=0

### 19.6.13 OCTOSPI communication configuration register (OCTOSPI\_CCR)

Address offset: 0x0100

Reset value: 0x0000 0000





- Bit 31 **SIOO**: Send instruction only once mode  
See [Section : Sending the instruction only once on page 540](#). This bit has no effect when IMODE=00  
0: Send instruction on every transaction  
1: Send instruction only for the first command  
This field can be written only when BUSY=0
- Bit 30 Reserved, must be kept at reset value
- Bit 29 **DQSE**: DQS enable  
This bit enables the data strobe management  
0: DQS disabled  
1: DQS enabled
- Bit 28 Reserved, must be kept at reset value
- Bit 27 **DDTR**: Alternate bytes double transfer rate  
This bit sets the DTR Mode for the alternate bytes phase:  
0: DTR Mode disabled for alternate bytes phase  
1: DTR Mode enabled for alternate bytes phase  
This field can be written only when BUSY=0
- Bits 26:24 **DMODE[2:0]**: Data mode  
This field defines the data phase's mode of operation:  
000: No instruction  
001: Data on a single line  
010: Data on two lines  
011: Data on four lines  
100: Data on eight lines  
101-111: Reserved, must be kept at reset value  
This field can be written only when BUSY=0
- Bits 23:22 Reserved, must be kept at reset value
- Bits 21:20 **ABSIZE[1:0]**: Alternate bytes size  
This bit defines alternate bytes size:  
00: 8-bit alternate bytes  
01: 16-bit alternate bytes  
10: 24-bit alternate bytes  
11: 32-bit alternate bytes  
This field can be written only when BUSY=0
- Bit 19 **ABDTR**: Alternate bytes double transfer rate  
This bit sets the DTR mode for the alternate bytes phase:  
0: DTR mode disabled for alternate bytes phase  
1: DTR mode enabled for alternate bytes phase  
This field can be written only when BUSY=0

Bits 18:16 **ABMODE[2:0]**: Alternate byte mode  
This field defines the alternate byte phase's mode of operation:  
000: No instruction  
001: Alternate bytes on a single line  
010: Alternate bytes on two lines  
011: Alternate bytes on four lines  
100: Alternate bytes on height lines  
101-111: Reserved, must be kept at reset value  
This field can be written only when BUSY=0

Bits 15:14 Reserved, must be kept at reset value

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  
This field can be written only when BUSY=0

Bit 11 **ADDTR**: Address double transfer rate  
This bit sets the DTR mode for the address phase:  
0: DTR mode disabled for address phase  
1: DTR mode enabled for address phase  
This field can be written only when BUSY=0

Bits 10:8 **ADMODE[2:0]**: Address mode  
This field defines the address phase's mode of operation:  
000: No address  
001: Address on a single line  
010: Address on two lines  
011: Address on four lines  
100: Address on height lines  
101-111: Reserved, must be kept at reset value  
This field can be written only when BUSY=0.

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

- Bits 5:4 **ISIZE[1:0]**: Instruction size  
 This bit defines instruction size:  
 00: 8-bit instruction  
 01: 16-bit instruction  
 10: 24-bit instruction  
 11: 32-bit instruction  
 This field can be written only when BUSY=0
- Bit 3 **IDTR**: Instruction double transfer rate  
 This bit sets the DTR Mode for the instruction phase:  
 0: DTR mode disabled for instruction phase  
 1: DTR mode enabled for instruction phase  
 This field can be written only when BUSY=0
- Bits 2:0 **IMODE[2:0]**: Instruction mode  
 This field defines the instruction phase's mode of operation:  
 000: No instruction  
 001: Instruction on a single line  
 010: Instruction on two lines  
 011: Instruction on four lines  
 100: Instruction on height lines  
 101-111: Reserved, must be kept at reset value  
 This field can be written only when BUSY=0

### 19.6.14 OCTOSPI timing configuration register (OCTOSPI\_TCR)

Address offset: 0x0108

Reset value: 0x0000 0000

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

- Bit 31 Reserved, must be kept at reset value
- Bit 30 **SSHIFT**: Sample shift  
 By default, the OCTOSPI samples data 1/2 of a CLK cycle after the data is driven by the external device  
 This bit allows the data to be sampled later in order to consider the external signal delays  
 0: No shift  
 1: 1/2 cycle shift  
 The firmware must assure that SSHIFT=0 when the data phase is configured in DTR mode (when DDTR=1)  
 This field can be modified only when BUSY=0
- Bit 29 Reserved, must be kept at reset value

Bit 28 **DHQC**: Delay hold quarter cycle

Bits 27: 5 Reserved, must be kept at reset value

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles

This field defines the duration of the dummy phase

In both SDR and DTR modes, it specifies a number of CLK cycles (0-31)

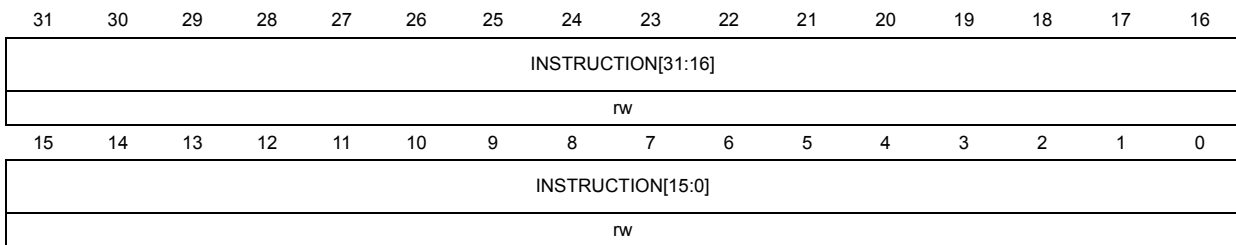
It is recommended to have at least five dummy cycles when using memories with DQS activated

This field can be written only when BUSY=0

### 19.6.15 OCTOSPI instruction register (OCTOSPI\_IR)

Address offset: 0x0110

Reset value: 0x0000 0000



Bits 31: 0 **INSTRUCTION[31: 0]**: Instruction

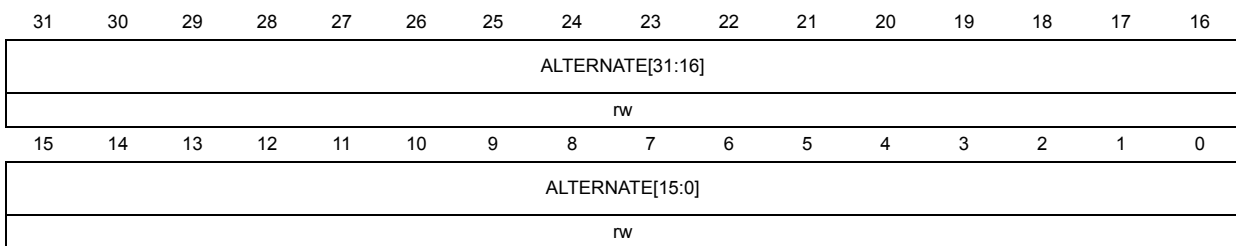
Instruction to be send to the external SPI device

This field can be written only when BUSY=0

### 19.6.16 OCTOSPI alternate bytes register (OCTOSPI\_ABR)

Address offset: 0x0120

Reset value: 0x0000 0000



Bits 31:0 **ALTERNATE[31: 0]**: Alternate bytes

Optional data to be send to the external SPI device right after the address.

This field can be written only when BUSY=0.

### 19.6.17 OCTOSPI low-power timeout register (OCTOSPI\_LPTR)

Address offset: 0x00130

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
TIMEOUT[15:0]															
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 OCTOSPI prefetches the subsequent bytes and hold them in the FIFO

This field indicates how many CLK cycles the OCTOSPI waits after the clock becomes inactive and until it raises the nCS, putting the external device in a lower-consumption state

This field can be written only when BUSY=0

### 19.6.18 OCTOSPI write communication configuration register (OCTOSPI\_WCCR)

Address offset: 0x0180

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.		DQSE	Res.	DDTR	DMODE			Res.		ABSIZE	ABDTR	ABMODE			
		rw		rw	rw					rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		ADSIZE		ADDT R	ADMODE			Res.		ISIZE	IDTR	IMODE			
		rw		rw	rw					rw	rw	rw			

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

Bit 29 **DQSE**: DQS enable

This bit enables the data strobe management

0: DQS disabled

1: DQS enabled

Bit 28 Reserved, must be kept at reset value

Bit 27 **DDTR**: data double transfer rate

This bit sets the DTR mode for the data phase:

0: DTR mode disabled for data phase

1: DTR mode enabled for data phase

This field can be written only when BUSY=0

Bits 26:24 **DMODE[2:0]**: Data mode

This field defines the data phase's mode of operation:

000: No instruction

001: Data on a single line

010: Data on two lines

011: Data on four lines

011: Data on four lines

100: Data on height lines

101-111: Reserved, must be kept at reset value

This field can be written only when BUSY=0

Bits 23:22 Reserved, must be kept at reset value

Bits 21:20 **ABSIZE[1:0]**: Alternate bytes size

This bit defines alternate bytes size:

00: 8-bit alternate bytes

01: 16-bit alternate bytes

10: 24-bit alternate bytes

11: 32-bit alternate bytes

This field can be written only when BUSY=0

Bit 19 **ABDTR**: Alternate bytes double transfer rate

This bit sets the DTR mode for the alternate bytes phase:

0: DTR mode disabled for alternate bytes phase

1: DTR mode enabled for alternate bytes phase

This field can be written only when BUSY=0

Bits 18:16 **ABMODE[2:0]**: Alternate byte mode

This field defines the alternate byte phase's mode of operation:

000: No instruction

001: Alternate bytes on a single line

010: Alternate bytes on two lines

011: Alternate bytes on four lines

011: Alternate bytes on four lines

100: Alternate bytes on height lines

101-111: Reserved, must be kept at reset value

This field can be written only when BUSY=0

Bits 15:14 Reserved, must be kept at reset value

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

This field can be written only when BUSY=0

Bit 11 **ADDTR**: Address double transfer rate

This bit sets the DTR Mode for the address phase:

0: DTR mode disabled for address phase

1: DTR mode enabled for address phase

This field can be written only when BUSY=0

Bits 10:8 **ADMODE[2:0]**: Address mode

This field defines the address phase's mode of operation:

000: No address

001: Address on a single line

010: Address on two lines

011: Address on four lines

011: Address on four lines

100: Address on height lines

101-111: Reserved, must be kept at reset value

This field can be written only when BUSY=0

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

Bits 5:4 **ISIZE[1:0]**: Instruction size

This bit defines instruction size:

00: 8-bit instruction

01: 16-bit instruction

10: 24-bit instruction

11: 32-bit instruction

This field can be written only when BUSY=0

Bit 3 **IDTR**: Instruction double transfer rate

This bit sets the DTR mode for the instruction phase:

0: DTR mode disabled for instruction phase

1: DTR mode enabled for instruction phase

This field can be written only when BUSY=0

Bits 2:0 **IMODE[2:0]**: Instruction mode

This field defines the instruction phase's mode of operation:

000: No instruction

001: Instruction on a single line

010: Instruction on two lines

011: Instruction on four lines

011: Instruction on four lines

100: Instruction on height lines

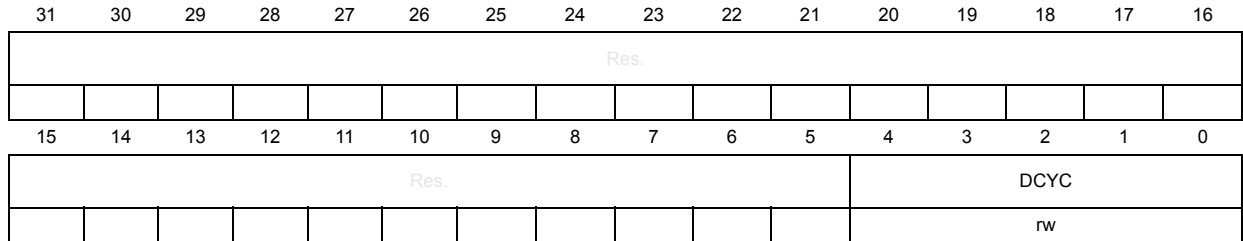
101-111: Reserved

This field can be written only when BUSY=0

**19.6.19 OCTOSPI write timing configuration register (OCTOSPI\_WTCR)**

Address offset: 0x0188

Reset value: 0x0000 0000



Bits 31:5 Reserved, must be kept at reset value

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles

This field defines the duration of the dummy phase

In both SDR and DTR modes, it specifies a number of CLK cycles (0-31)

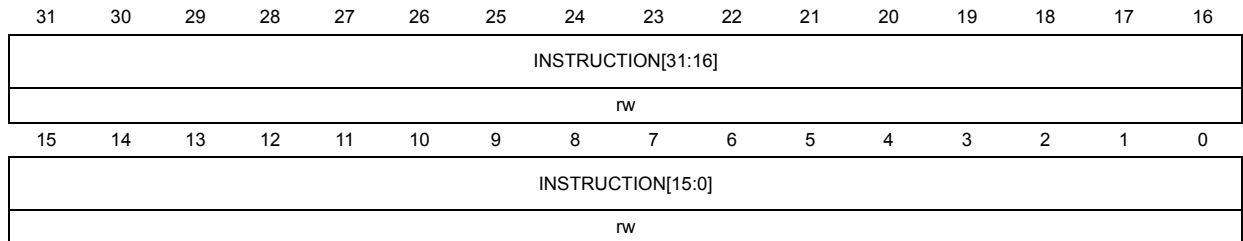
It is recommended to have at least 5 dummy cycles when using memories with DQS activated

This field can be written only when BUSY=0

**19.6.20 OCTOSPI write instruction register (OCTOSPI\_WIR)**

Address offset: 0x0190

Reset value: 0x0000 0000



Bits 31:0 **INSTRUCTION[31: 0]**: Instruction

Instruction to be send to the external SPI device

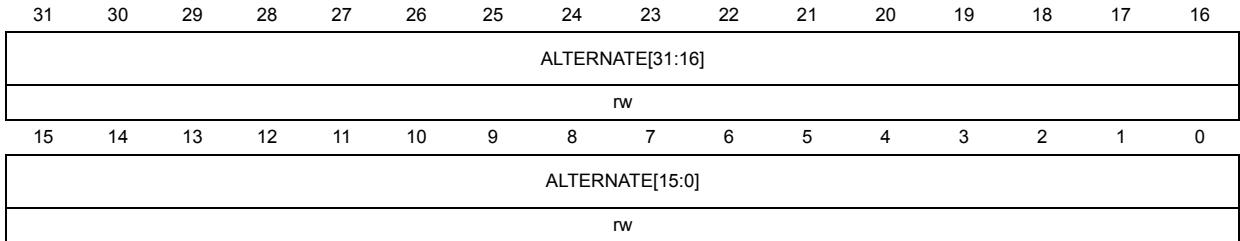
This field can be written only when BUSY=0



### 19.6.21 OCTOSPI write alternate bytes register (OCTOSPI\_WABR)

Address offset: 0x01A0

Reset value: 0x0000 0000

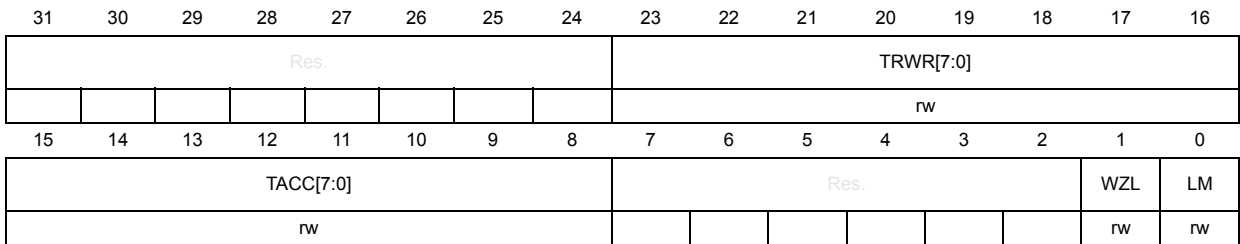


Bits 31:0 **ALTERNATE[31: 0]**: Alternate bytes  
 Optional data to be sent to the external SPI device right after the address  
 This field can be written only when BUSY=0

### 19.6.22 OCTOSPI HyperBus™ latency configuration register (OCTOSPI\_HLCR)

Address offset: 0x0200

Reset value: 0x0000 0000



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

Bits 23:16 **TRWR**: Read write recovery time  
 Device read write recovery time expressed in number of communication clock cycles

Bits 15:8 **TACC[7: 0]**: Access time  
 Device access time expressed in number of communication clock cycles

Bits 7:2 Reserved, must be kept at reset value

Bit 1 **WZL**: Write zero latency  
 This bit enables zero latency on write operations  
 0: Latency on write accesses  
 1: No latency on write accesses

Bit 0 **LM**: Latency mode  
 This bit select the latency mode  
 0: Variable initial latency  
 1: Fixed latency



### 19.6.23 OCTOSPI register map

The following table summarizes the OCTOSPI registers. Refer to the register boundary addresses table for the OCTOSPI register base address.

**Table 112. OCTOSPI register map**

Offset	Register name reset value	Register size																																	
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0x0000	OCTOSPI_CR	Res.	Res.	FMDOE[1:0]		Res.	Res.	Res.	Res.	Res.	PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE	Res.	Res.	Res.	FTHRES[4:0]				FSEL	DOM	Res.	Res.	Res.	TCEN	DMAEN	ABORT.	EN.	
	Reset value			0	0						0	0		0	0	0	0	0				0	0	0	0	0	0	0	0		0	0			
0x0004	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.		
0x0008	OCTOSPI_DCR1	Res.	Res.	Res.	Res.	Res.	MTYP [2:0]		Res.	Res.	Res.	DEVSIZ[4:0]				Res.	Res.	Res.	Res.	Res.	Res.	CSHT [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRCK.	CKMODE.		
	Reset value						0	0	0					0	0	0	0	0					0	0	0							0	0		
0x000C	OCTOSPI_DCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRAPSIZE[2:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[7:0]						
	Reset value															0	0	0											0	0	0	0	0	0	0
0x0010	OCTOSPI_DCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSBOUND[4:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value													0	0	0	0																		
0x0014 - 0x001C	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.		
0x0020	OCTOSPI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLEVEL[5:0]					Res.	Res.	BUSY	TOF	SMF	FTF	TCF	TEF	
	Reset value																						0	0	0	0	0	0	0	0	0	0	0		
0x0024	OCTOSPI_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTOF	CSMF	CTCF	CTEF		
	Reset value																												0	0	0	0			
0x0028 - 0x003C	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.		
0x0040	OCTOSPI_DLR	DL[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		
0x0044	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.		
0x0048	OCTOSPI_AR	ADDRESS[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		
0x004C	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.		



Table 112. OCTOSPI register map (continued)

Offset	Register name reset value	Register size																																
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x0050	OCTOSPI_DR	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	0	0
0x0054 - 0x007C	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.
0x0080	OCTOSPI_PSMKR	MASK[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	
0x0084	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.	
0x0088	OCTOSPI_PSMAR	MATCH[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	
0x008C	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.	
0x0090	OCTOSPI_PIR	RESERVED															INTERVAL[15:0]																	
	Reset value																																	
0x0094 - 0x00FC	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.	
0x0100	OCTOSPI_CCR	SIOO	Res.	Res.	DQSE	Res.	DDTR	Res.	DMODE [2:0]	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																										
0x0104	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.		
0x0108	OCTOSPI_TCR	Res.	SSHIFT	Res.	Res.	DHQC	Res.	Res.	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																												
0x010C	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.		
0x0110	OCTOSPI_IR	INSTRUCTION[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	
0x0114 - 0x011C	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.	
0x0120	OCTOSPI_ABR	ALTERNATE[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	
0x0124 - 0x012C	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.	



Table 112. OCTOSPI register map (continued)

Offset	Register name reset value	Register size																																		
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
0x0130	OCTOSPI_LPTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	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	
0x0134 - 0x017C	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		
0x0180	OCTOSPI_WCCR	Res	Res	Res	DQSE	Res	DDTR	DMODE [2:0]		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		
	Reset value	0		0		0	0	0	0	0			0	0	0	0	0																			
0x0184	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		
0x0188	OCTOSPI_WTCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	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																																			
0x018C	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		
0x0190	OCTOSPI_WIR	INSTRUCTION[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	
0x0194 - 0x019C	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		
0x01A0	OCTOSPI_WABR	ALTERNATE[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	
0x01A4 - 0x01FC	Reserved	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	
0x0200	OCTOSPI_HLCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	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



## 20 OctoSPI IO Manager (OCTOSPIM)

### 20.1 Introduction

The OctoSPI IO Manager is a low level interface enabling

- Efficient OctoSPI pin assignment with a full IO Matrix (before alternate function map)

### 20.2 OctoSPI IO Manager main features

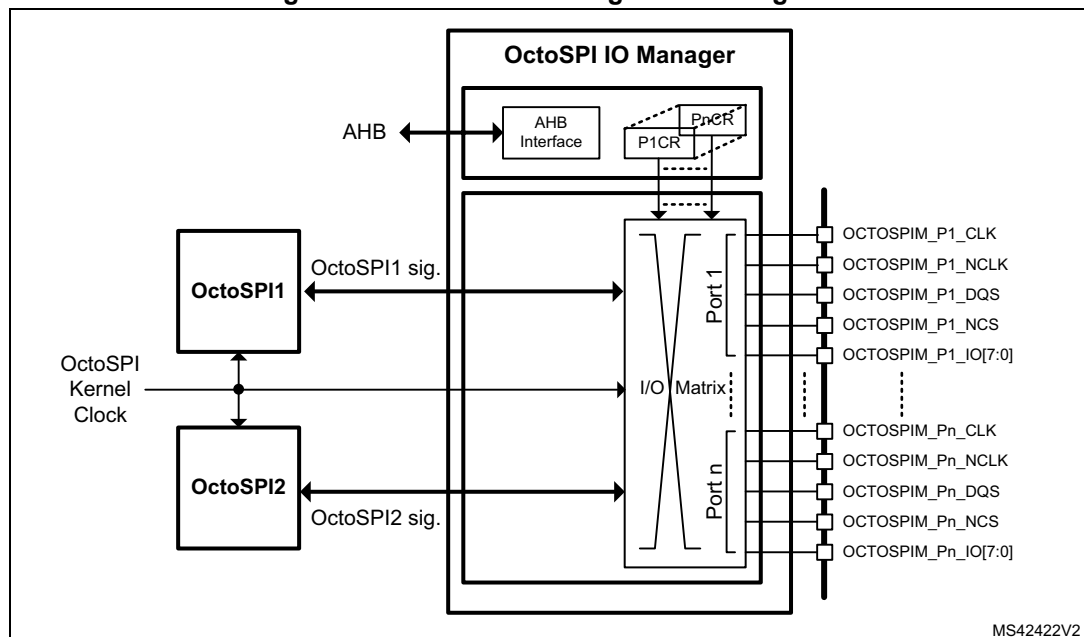
- Supports up to two Single/Dual/Quad/Octal SPI Interfaces
- Supports up to 3 ports for pin assignment
- Fully programmable IO Matrix for pin assignment by function (data/control/clock)

### 20.3 OctoSPI IO Manager functional description

#### 20.3.1 General description

The block diagram of the OctoSPI IO Manager is shown in [Figure 83](#).

Figure 83. OctoSPI IO Manager block diagram



1. The number of ports (n) is in the range from 1 to 3.

### 20.3.2 OctoSPI IO Manager Matrix

The OctoSPI IO Manager Matrix allows the user to set a fully programmable pre-mapping of functions:

- Any OCTOSPIM\_Pn\_CLK / OCTOSPIM\_Pn\_NCLK pair can be mapped independently to OctoSPI1\_CLK/OctoSPI1\_NCLK or OctoSPI2\_CLK/OctoSPI2\_NCLK
- Any OCTOSPIM\_Pn\_DQS can be mapped independently to OctoSPI1\_DQS or OctoSPI2\_DQS
- Any OCTOSPIM\_Pn\_NCS can be mapped independently to OctoSPI1\_NCS or OctoSPI2\_NCS
- Any OCTOSPIM\_Pn\_IO[3:0] and OCTOSPIM\_Pn\_IO[7:4] can be mapped independently to OctoSPI1\_IO[3:0], OctoSPI1\_IO[7:4], OctoSPI2\_IO1[3:0] or OctoSPI2\_IO[7:4]

For each OctoSPI IO Manager port, individual signal enables and mapping are configured through the corresponding OctoSPI IO manager Port n configuration register (OCTOSPIM\_PnCR).

When several I/O pins have the same configuration, and are enabled in the same time, the result may be not predictable.

In the default Out of reset configuration, all the signals of the OctoSPI1 and OctoSPI2 are mapped, respectively, on Port 1 and on Port 2.

The I/O Matrix shall be configured when the OctoSPIs are disabled, to avoid unexpected transactions on the bus.

## 20.4 OctoSPI IO Manager registers

### 20.4.1 OctoSPI IO Manager Port n configuration register (OCTOSPIM\_PnCR)

Address offset: 0x0000 + 4 x n

Reset value: 0x0301 0111 for n = 1

Reset value: 0x0705 0333 for n = 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	IOHSRC[1:0]		IOHEN	Res.	Res.	Res.	Res.	Res.	IOLSRC[1:0]		IOLEN
					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.	NCSSRC	NCSSEN	Res.	Res.	DQSSRC	DQSEN	Res.	Res.	CLKSRC	CLKEN
						rw	rw			rw	rw			rw	rw

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

Bits 26:25 **IOHSRC[1:0]**: IO[7:4] Source for Port n

This bits select the source of Port n IO[7:4].

00: OCTOSPI1\_IO[3:0]

01: OCTOSPI1\_IO[7:4]

10: OCTOSPI2\_IO[3:0]

11: OCTOSPI2\_IO[7:4]

Bit 24 **IOHEN**: IO[7:4] Enable for Port n

This bit enables the Port n IO[7:4].

0: IO[7:4] for Port n is disabled

1: IO[7:4] for Port n is enabled

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

Bits 18:17 **IOLSRC[1:0]**: IO[3:0] Source for Port n

This bits select the source of Port n IO[3:0].

00: OCTOSPI1\_IO[3:0]

01: OCTOSPI1\_IO[7:4]

10: OCTOSPI2\_IO[3:0]

11: OCTOSPI2\_IO[7:4]

Bit 16 **IOLEN**: IO[3:0] Enable for Port n

This bit enables the Port n IO[3:0].

0: IO[3:0] for Port n is disabled

1: IO[3:0] for Port n is enabled

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **NCSSRC**: nCS Source for Port n

This bit selects the source of Port n nCS.

0: OCTOSPI1\_nCS

1: OCTOSPI2\_nCS



Bit 8 **NCSSEN**: nCS Enable for Port n  
This bit enables the Port n nCS.  
0: nCS for Port n is disabled  
1: nCS for Port n is enabled

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

Bit 5 **DQSSRC**: DQS Source for Port n  
This bit selects the source of Port n DQS.  
0: OCTOSPI1\_DQS  
1: OCTOSPI2\_DQS

Bit 4 **DQSEN**: DQS Enable for Port n  
This bit enables the Port n DQS.  
0: DQS for Port n is disabled  
1: DQS for Port n is enabled

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

Bit 1 **CLKSRC**: CLK/CLKn Source for Port n  
This bit selects the source of Port n CLK/CLKn.  
0: OCTOSPI1\_CLK/CLKn  
1: OCTOSPI2\_CLK/CLKn

Bit 0 **CLKEN**: CLK/CLKn Enable for Port n  
This bit enables the Port n CLK/CLKn.  
0: CLK/CLKn for Port n is disabled  
1: CLK/CLKn for Port n is enabled

### 20.4.2 OctoSPI IO Manager register map

The following table summarizes the OctoSPI IO Manager registers. Refer to the register boundary addresses table for the OctoSPI IO Manager register base address.

**Table 113. OCTOSPIM IO Manager 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
0x0004	OCTOSPIM_P1CR	Res.	Res.	Res.	Res.	Res.	IOHSRC	IOHSRC	IOHEN	Res.	Res.	Res.	Res.	Res.	Res.	IOLSRC	IOLEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NCSSRC	NCSEN	Res.	Res.	DQSSRC	DQSEN	Res.	Res.	CLKSRC	CLKEN
	Reset value						0	1	1							0	1								0	1			0	1			0	1
0x0008	OCTOSPIM_P2CR	Res.	Res.	Res.	Res.	Res.	IOHSRC	IOHSRC	IOHEN	Res.	Res.	Res.	Res.	Res.	Res.	IOLSRC	IOLEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NCSSRC	NCSEN	Res.	Res.	DQSSRC	DQSEN	Res.	Res.	CLKSRC	CLKEN
	Reset value						1	1	1							1	0								1	1			1	1			1	1



## 21 Analog-to-digital converters (ADC)

### 21.1 Introduction

The ADC consists of a 12-bit successive approximation analog-to-digital converter.

The ADC has up to 20 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 performances 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.

## 21.2 ADC main features

- High-performance features
  - 12, 10, 8 or 6-bit configurable resolution
  - ADC conversion time:
    - Fast channels: 0.188  $\mu$ s for 12-bit resolution (5.33 Ms/s)
    - Slow channels: 0.238  $\mu$ s for 12-bit resolution (4.21 Ms/s)
  - ADC conversion time is independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution: 0.16  $\mu$ s for 10-bit 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 GP-DMA for regular channel conversions
  - Data can be routed to DFSDM for post processing
  - 4 dedicated data registers for the injected channels
- Oversampler
  - 16-bit data register
  - Oversampling ratio adjustable from 2 to 256x
  - 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 (0.188  $\mu$ s conversion time for fast channels can be kept whatever the AHB bus clock frequency)
  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency application (auto-delayed mode)
- Several external analog input channel
  - 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 ( $V_{REFINT}$ )
  - The internal temperature sensor ( $V_{SENSE}$ )
  - The  $V_{BAT}$  monitoring channel ( $V_{BAT}/3$ )
  - DAC1 and DAC2 internal channels, 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 84* shows the block diagram of one ADC.

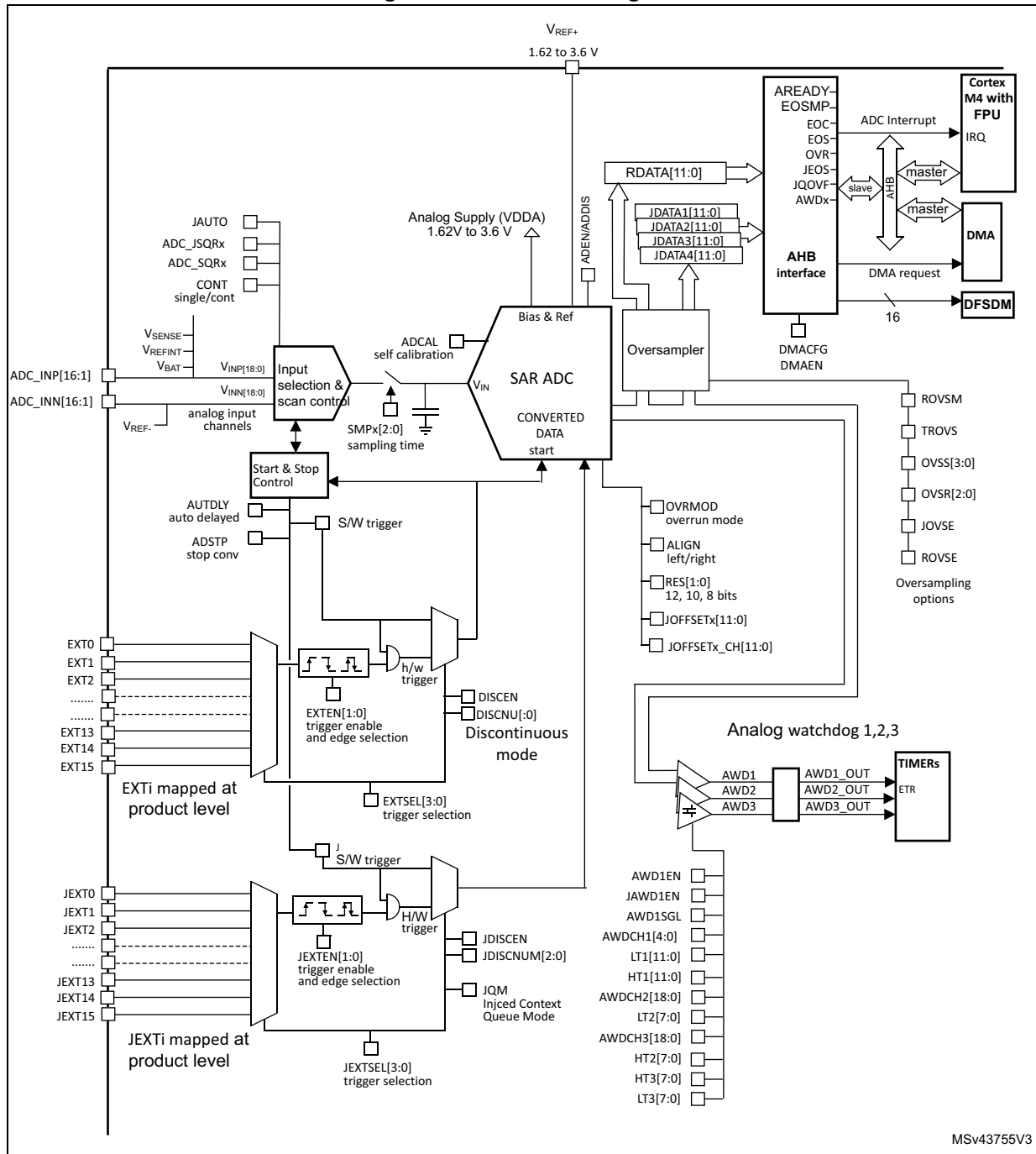
## 21.3 ADC implementation

## 21.4 ADC functional description

### 21.4.1 ADC block diagram

[Figure 84](#) shows the ADC block diagram and [Table 115](#) gives the ADC pin description.

Figure 84. ADC block diagram



MSv43755V3

## 21.4.2 ADC pins and internal signals

Table 114. 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_AWDx_OUT	Output	Internal analog watchdog output signal connected to on-chip timers. (x = Analog watchdog number 1,2,3)
V <sub>SENSE</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 115. ADC input/output pins

Pin name	Signal type	Comments
V <sub>REF+</sub>	Input, analog reference positive	The higher/positive reference voltage for the ADC, $1.62\text{ V} \leq V_{\text{REF}+} \leq V_{\text{DDA}}$
V <sub>DDA</sub>	Input, analog supply	Analog power supply equal V <sub>DDA</sub> : $1.62\text{ V} \leq V_{\text{DDA}} \leq 3.6\text{ V}$
V <sub>REF-</sub>	Input, analog reference negative	The lower/negative reference voltage for the ADC. V <sub>REF-</sub> is internally connected to V <sub>SSA</sub>
V <sub>SSA</sub>	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>INP</sub> [18:0]	Positive analog input channels	Connected either to external channels: ADC1_INP <i>i</i> or internal channels.
V <sub>INN</sub> [18:0]	Negative analog input channels	Connected to V <sub>REF-</sub> or external channels: ADC1_INN <i>i</i> and ADC1_INP[ <i>i</i> +1]
ADC1_INN[16:1]	External analog input signals	Up to 16 analog input channels : – 5 fast channels (ADC1_INN1 to INN5) – Up to 11 slow channels (ADC1_INN6 to INN16)
ADC1_INP[16:1]	External analog input signals	Up to 16 analog input channels : – 5 fast channels (ADC1_INP1 to INP5) – Up to 11 slow channels (ADC1_INP6 to INP16)



### 21.4.3 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 the same for the three ADCs and can be selected between two different clock sources (see [Figure 85: ADC clock scheme](#)):

1. The ADC clock can be a specific clock source. It can be 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]).

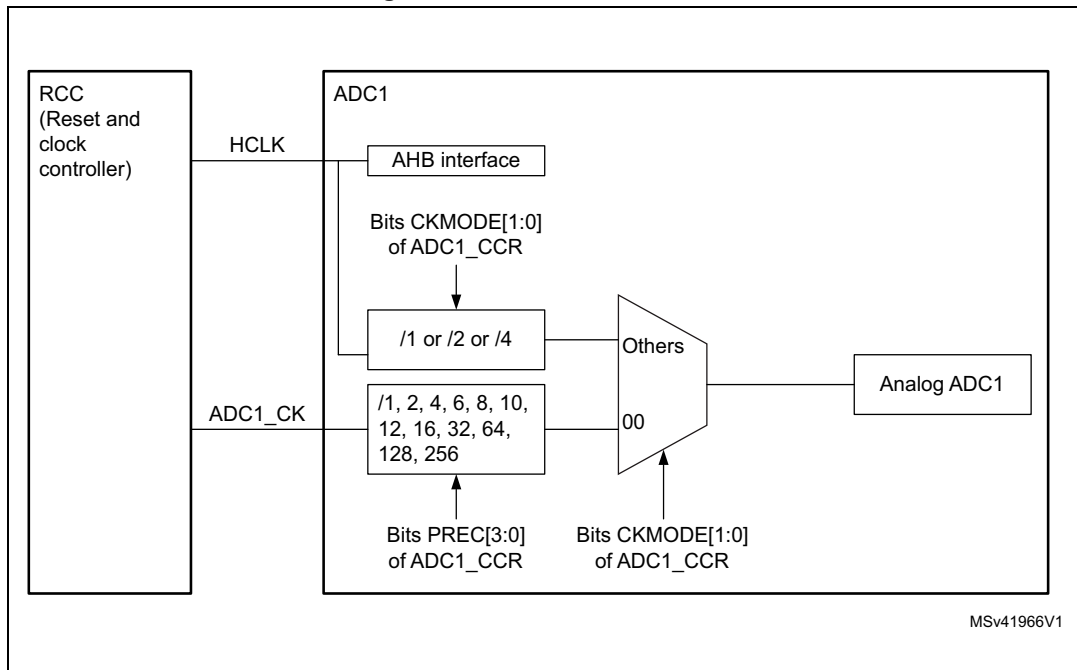
To select this scheme, bits CKMODE[1:0] of the ADCx\_CCR register must be different from "00".

*Note:* For option 2), a prescaling factor of 1 (CKMODE[1:0]=01) can be used only if the AHB prescaler is set to 1 (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 is added by the resynchronizations between the two clock domains).

Figure 85. ADC clock scheme



MSv41966V1

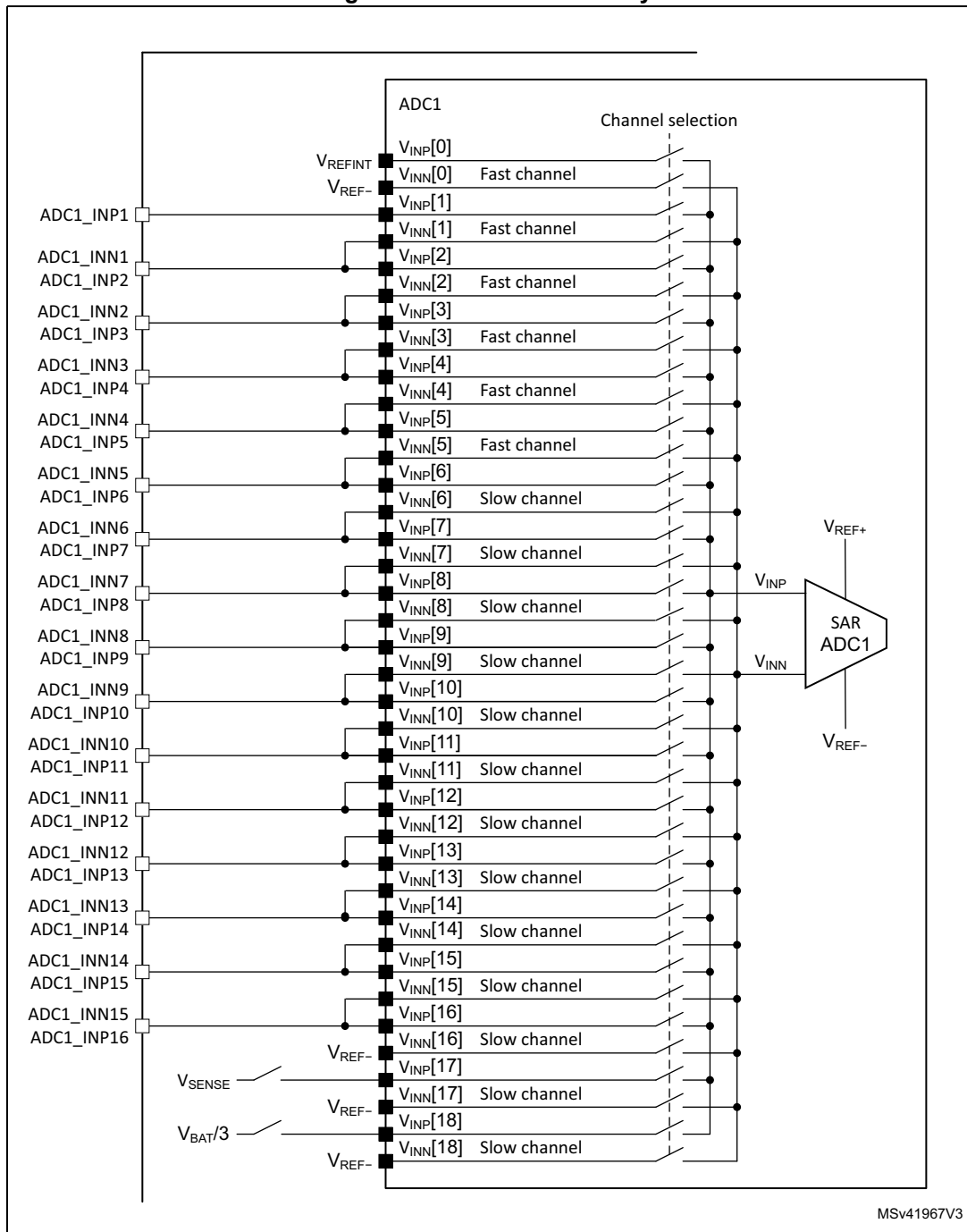
**Clock ratio constraint between ADC clock and AHB clock**

There are generally no constraints to be respected for the ratio between the ADC clock and the AHB clock except if some injected channels are programmed. In this case, it is mandatory to respect the following ratio:

- $F_{HCLK} \geq F_{ADC} / 4$  if the resolution of all channels are 12-bit or 10-bit
- $F_{HCLK} \geq F_{ADC} / 3$  if there are some channels with resolutions equal to 8-bit (and none with lower resolutions)
- $F_{HCLK} \geq F_{ADC} / 2$  if there are some channels with resolutions equal to 6-bit

21.4.4 ADC1 connectivity

Figure 86. ADC1 connectivity



### 21.4.5 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.

### 21.4.6 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 ADCx\_CR register).

To start ADC operations, it is first needed to exit Deep-power-down mode by setting bit DEEPPWD=0.

Then, it is mandatory to enable the ADC internal voltage regulator by setting the bit ADVREGEN=1 into ADCx\_CR register. The software must wait for the startup time of the ADC voltage regulator ( $T_{\text{ADCVREG\_STUP}}$ ) before launching a calibration or enabling the ADC. This delay must be implemented by software.

For the startup time of the ADC voltage regulator, refer to device datasheet for  $T_{\text{ADCVREG\_STUP}}$  parameter.

After 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 ADCx\_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 21.4.8: Calibration \(ADCAL, ADCALDIF, ADCx\\_CALFACT\)](#)).

### 21.4.7 Single-ended and differential input channels

Channels can be configured to be either single-ended input or differential input by writing into bits DIFSEL[15:1] in the ADCx\_DIFSEL register. This configuration must be written while the ADC is disabled (ADEN=0). Note that DIFSEL[18:16,0] are fixed to single ended channels and are always read as 0.

In single-ended input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage  $V_{INP[i]}$  (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 external voltage  $V_{INP[i]}$  (positive input) and  $V_{INN[i]}$  (negative input).

When ADC is configured as differential mode, both inputs should be biased at  $(V_{REF+}) / 2$  voltage.

The input signal are supposed to be differential (common mode voltage should be fixed).

For a complete description of how the input channels are connected, refer to [Section 21.4.4: ADC1 connectivity](#).

**Caution:** When configuring the channel “i” in differential input mode, its negative input voltage  $V_{INN[i]}$  is connected to  $V_{INP[i+1]}$ . As a consequence, channel “i+1” is no longer usable in single-ended mode or in differential mode and must never be configured to be converted.

**Note:** ADC channels 0, 16, 17, 18 are forced to single-ended configuration (corresponding bits DIFSEL[j] is always zero), either because connected to a single-ended external analog input or connected to an internal channel.

### 21.4.8 Calibration (ADCAL, ADCALDIF, ADCx\_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 which will be applied for single-ended input conversions.
- Write ADCALDIF=1 before launching a calibration which will be applied for differential input conversions.

The calibration is then initiated by software by setting bit ADCAL=1. Calibration can only be initiated when the ADC is disabled (when ADEN=0). ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT\_S[6:0] or CALFACT\_D[6:0] of ADCx\_CALFACT register (depending on single-ended or differential input calibration)

The internal analog calibration is kept if the ADC is disabled (ADEN=0). However, if the ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before re-enabling the ADC.

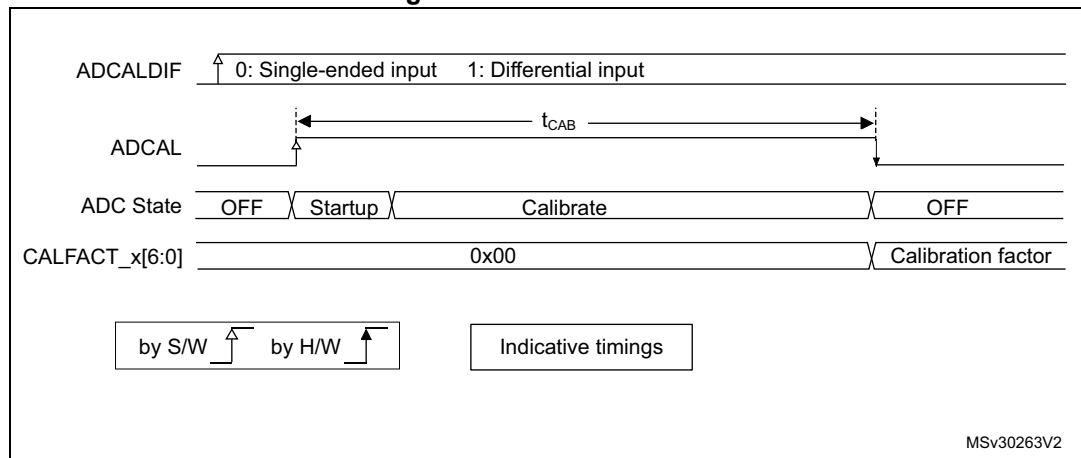
The internal analog calibration is lost each time the power of the ADC is removed (example, when the product enters in STANDBY or VBAT mode). In this case, to avoid spending time recalibrating the ADC, it is possible to re-write the calibration factor into the ADCx\_CALFACT register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

The calibration factor can be written if the ADC is enabled but not converting (ADEN=1 and ADSTART=0 and JADSTART=0). Then, at the next start of conversion, the calibration factor will automatically be injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion. It is recommended to recalibrate when V<sub>REF+</sub> 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=1.
5. Wait until ADCAL=0.
6. The calibration factor can be read from ADCx\_CALFACT register.

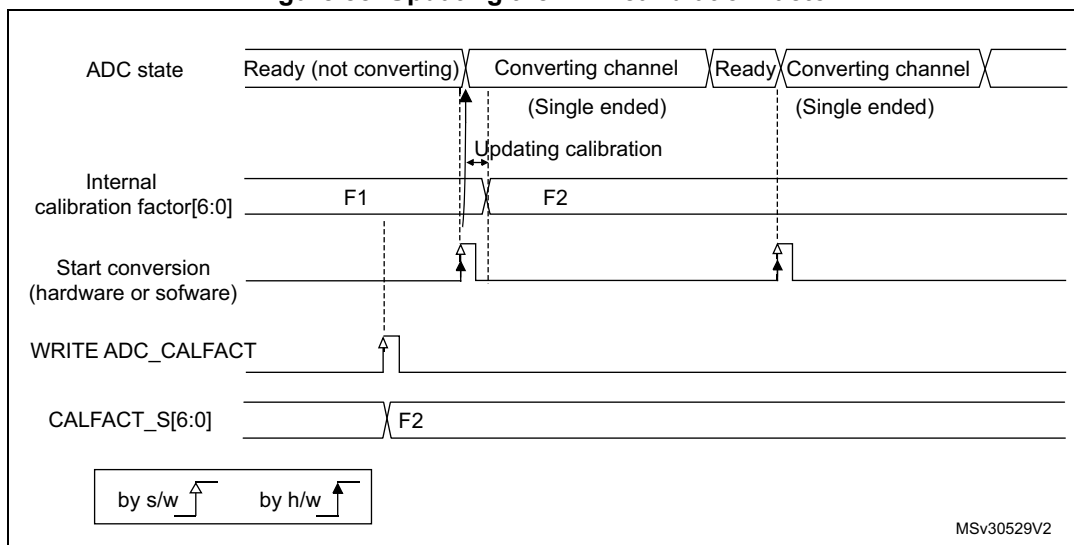
**Figure 87. ADC calibration**



**Software procedure to re-inject a calibration factor into the ADC**

1. Ensure ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing).
2. Write CALFACT\_S and CALFACT\_D with the new calibration factors.
3. When a conversion is launched, the calibration factor will be injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits CALFACT\_S for single-ended input channel or bits CALFACT\_D for differential input channel.

Figure 88. Updating the ADC calibration factor

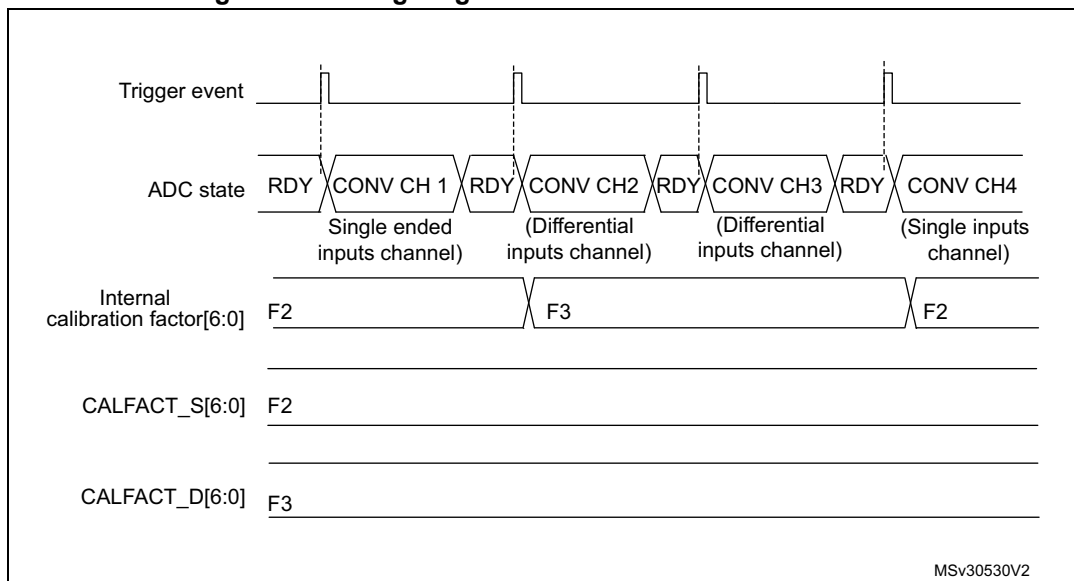


**Converting single-ended and differential analog inputs with a single ADC**

If the ADC is supposed to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF=0 and one with ADCALDIF=1. The procedure is the following:

1. Disable the ADC.
2. Calibrate the ADC in single-ended input mode (with ADCALDIF=0). This updates the register CALFACT\_S[6:0].
3. Calibrate the ADC in differential input modes (with ADCALDIF=1). This updates the register CALFACT\_D[6:0].
4. Enable the ADC, configure the channels and launch the conversions. Each time there is a switch from a single-ended to a differential inputs channel (and vice-versa), the calibration will automatically be injected into the analog ADC.

Figure 89. Mixing single-ended and differential channels



### 21.4.9 ADC on-off control (ADEN, ADDIS, ADRDY)

First of all, follow the procedure explained in [Section 21.4.6: 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 90](#). Two control bits enable or disable the ADC:

- ADEN=1 enables the ADC. The flag ADRDY will be set once the ADC is ready for operation.
- ADDIS=1 disables the ADC. 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 21.4.18: Conversion on external trigger and trigger polarity \(EXTSEL, EXTEN, JEXTSEL, JEXTEN\)](#)) or when an external trigger event occurs, if triggers are enabled.

Injected conversions start by setting JADSTART=1 or when an external injected trigger event occurs, if injected triggers are enabled.

#### Software procedure to enable the ADC

1. Clear the ADRDY bit in the ADCx\_ISR register by writing '1'.
2. Set ADEN=1.
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 ADCx\_ISR register by writing '1' (optional).

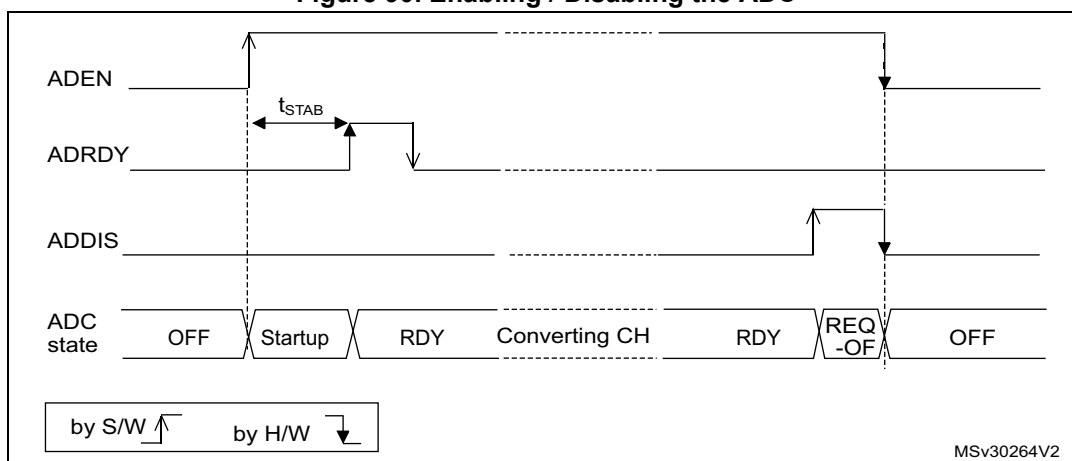
**Caution:** ADEN bit cannot be set during ADCAL=1 and 4 ADC clock cycle after the ADCAL bit is cleared by hardware (end of the calibration).

#### Software procedure to disable the ADC

1. Check that both ADSTART=0 and JADSTART=0 to ensure that no conversion is ongoing. If required, stop any regular and injected conversion ongoing by setting ADSTP=1 and JADSTP=1 and then wait until ADSTP=0 and JADSTP=0.
2. Set ADDIS=1.
3. If required by the application, wait until ADEN=0, until the analog ADC is effectively disabled (ADDIS will automatically be reset once ADEN=0).



Figure 90. Enabling / Disabling the ADC



### 21.4.10 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 control bits DIFSEL in the ADCx\_DIFSEL register and the control bits ADCAL and ADEN in the ADCx\_CR register, only if the ADC is disabled (ADEN must be equal to 0).

The software is then allowed to write the control bits ADSTART, JADSTART and ADDIS of the ADCx\_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

For all the other control bits of the ADCx\_CFGR, ADCx\_SMPRx, ADCx\_SQRY, ADCx\_JDRy, ADCx\_OFRRy, ADCx\_OFCHRY and ADCx\_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if the ADC is enabled (ADEN=1) and if there is no regular conversion ongoing (ADSTART must be equal to 0).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if the ADC is enabled (ADEN=1) and if there is no injected conversion ongoing (JADSTART must be equal to 0).

The software is allowed to write the ADSTP or JADSTP control bits of the ADCx\_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 ADCx\_JSQR at any time, when the ADC is enabled (ADEN=1). Refer to [Section 21.6.16: 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=0 as well as all the bits of ADCx\_CR register).

### 21.4.11 Channel selection (SQRx, JSQRx)

There are up to 20 multiplexed channels:

- 5 fast analog inputs coming from GPIO pads (ADC1\_INP/INN[1..5])
- 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/INN0.
  - the internal temperature sensor ( $V_{SENSE}$ ) is connected to ADC1\_INP17/INN17.
  - the  $V_{BAT}$  monitoring channel ( $V_{BAT/3}$ ) is connected to ADC1\_INP18/INN18.

*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\_INN2, 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 *ADCx\_SQRy* registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the *ADCx\_SQR1* register.
- An **injected group** is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the *ADCx\_JSQR* register. The total number of conversions in the injected group must be written in the L[1:0] bits in the *ADCx\_JSQR* register.

*ADCx\_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 21.4.17: Stopping an ongoing conversion \(ADSTP, JADSTP\)](#)).

It is possible to modify the *ADCx\_JSQR* registers on-the-fly while injected conversions are occurring. Refer to [Section 21.4.21: Queue of context for injected conversions](#)

### 21.4.12 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 ADCx\_SMPR1 and ADCx\_SMPR2 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:

$$T_{CONV} = \text{Sampling time} + 12.5 \text{ ADC clock cycles}$$

Example:

With  $F_{ADC\_CLK} = 80 \text{ MHz}$  and a sampling time of 2.5 ADC clock cycles:

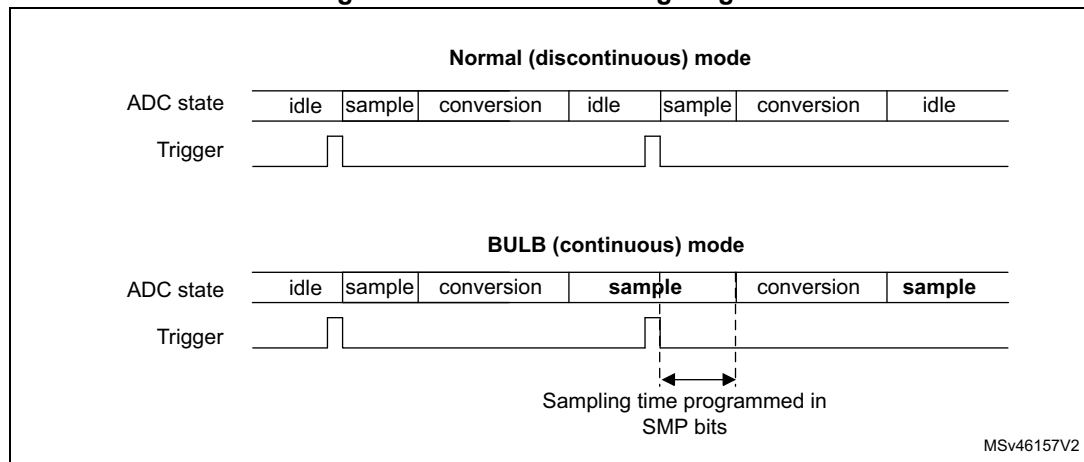
$$T_{CONV} = (2.5 + 12.5) \text{ ADC clock cycles} = 15 \text{ ADC clock cycles} = 187.5 \text{ ns (for fast channels)}$$

The ADC notifies the end of the sampling phase by setting the status bit EOSMP (only for regular conversion).

#### Constraints on the sampling time for fast and slow channels

For each channel, SMP[2:0] bits must be programmed to respect a minimum sampling time as specified in the ADC characteristics section of the datasheets.

Figure 91. Bulb mode timing diagram



### I/O analog switches voltage booster

The I/O analog switches resistance increases when the  $V_{DDA}$  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  $V_{DDA}$  by enabling an internal voltage booster with BOOSTEN bit in the SYSCFG\_CFGR1 register.

#### 21.4.13 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 ADCx\_CR register (for a regular channel)
- Setting the JADSTART bit in the ADCx\_CR register (for an injected channel)
- External hardware trigger event (for a regular or injected channel)

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADCx\_DR register
- The EOC (end of regular conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 16-bit ADCx\_JDRy registers
- The JEOC (end of injected conversion) flag is set
- An interrupt is generated if the JEOCIE bit is set

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set
- An interrupt is generated if the JEOSIE bit is set

Then the ADC stops until a new external regular or injected trigger occurs or until bit ADSTART or JADSTART is set again.

*Note:* To convert a single channel, program a sequence with a length of 1.

#### 21.4.14 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 ADCx\_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADCx\_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

*Note:* To convert a single channel, program a sequence with a length of 1.

*It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN=1 and CONT=1.*

*Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit), refer to [Auto-injection mode](#) section).*

### 21.4.15 Starting conversions (ADSTART, JADSTART)

Software starts ADC regular conversions by setting ADSTART=1.

When ADSTART is set, the conversion starts:

- Immediately: if EXTEN = 0x0 (software trigger)
- At the next active edge of the selected regular hardware trigger: if EXTEN != 0x0

Software starts ADC injected conversions by setting JADSTART=1.

When JADSTART is set, the conversion starts:

- Immediately, if JEXTEN = 0x0 (software trigger)
- At the next active edge of the selected injected hardware trigger: if JEXTEN != 0x0

*Note:* In auto-injection mode (JAUTO=1), use ADSTART bit to start the regular conversions followed by the auto-injected conversions (JADSTART must be kept cleared).

ADSTART and JADSTART also provide information on whether any ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART=0 and JADSTART=0 are both true, indicating that the ADC is idle.

ADSTART is cleared by hardware:

- In single mode with software regular trigger (CONT=0, EXTSEL=0x0)
  - at any end of regular conversion sequence (EOS assertion) or at any end of 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.

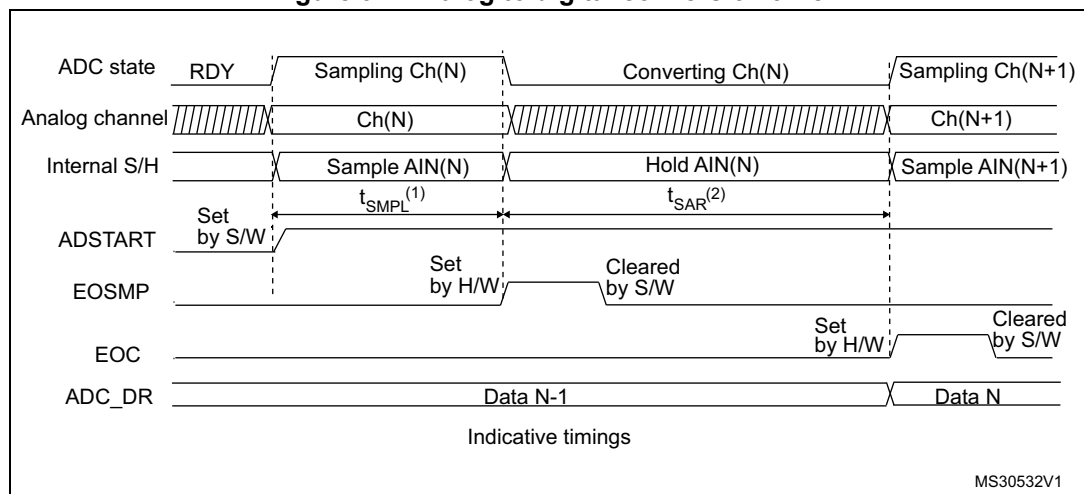
### 21.4.16 ADC timing

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:

$$T_{CONV} = T_{SMPL} + T_{SAR} = [ 2.5_{|min} + 12.5_{|12bit} ] \times T_{ADC\_CLK}$$

$$T_{CONV} = T_{SMPL} + T_{SAR} = 31.25 \text{ ns}_{|min} + 156.25 \text{ ns}_{|12bit} = 187.5 \text{ ns (for } F_{ADC\_CLK} = 80 \text{ MHz)}$$

**Figure 92. Analog to digital conversion time**



1.  $T_{SMPL}$  depends on SMP[2:0]
2.  $T_{SAR}$  depends on RES[2:0]

### 21.4.17 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 will reset the ongoing ADC operation. Then the ADC can be reconfigured (ex: changing the channel selection or the trigger) ready for a new operation.

Note that it is possible to stop injected conversions while regular conversions are still operating and vice-versa. This allows, for instance, re-configuration of the injected conversion sequence and triggers while regular conversions are still operating (and vice-versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with partial result discarded (ADCx\_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (ADCx\_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC would 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 93. Stopping ongoing regular conversions**

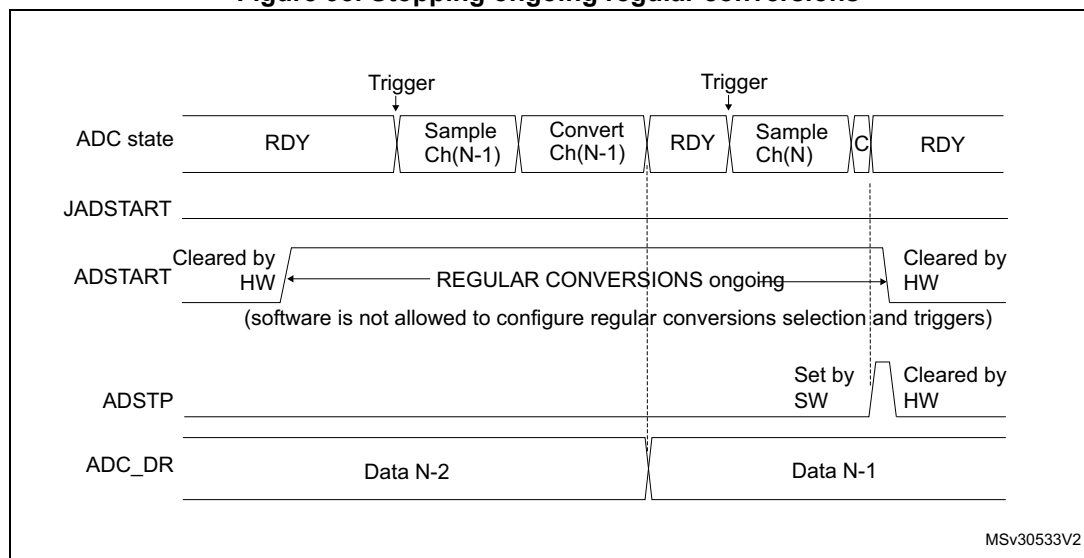
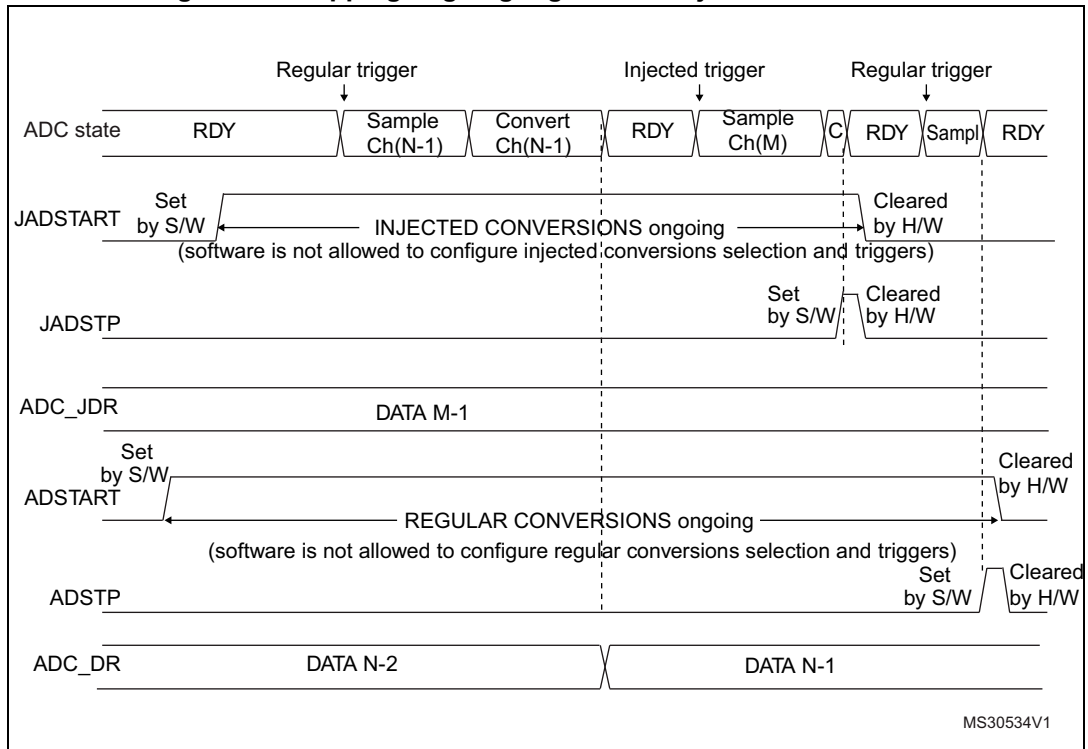


Figure 94. Stopping ongoing regular and injected conversions





### 21.4.18 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 0b00, 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 116](#) provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

**Table 116. 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 117. Configuring the trigger polarity for injected external triggers**

JEXTEN[1:0]	Source
00	– If JQDIS=1 (Queue disabled): Hardware trigger detection disabled, software trigger detection enabled – If JQDIS=0 (Queue enabled), Hardware and software trigger detection disabled
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 21.4.21: 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 21.4.21: Queue of context for injected conversions on page 599](#)

[Table 118](#) to [Table 119](#) give all the possible external triggers of the three ADCs for regular and injected conversion.

**Table 118. ADC1 - External triggers for regular channels**

Name	Source	Type	EXTSEL[3:0]
EXT0	TIM1_CH1	Internal signal from on-chip timers	0000
EXT1	TIM1_CH2	Internal signal from on-chip timers	0001
EXT2	TIM1_CH3	Internal signal from on-chip timers	0010
EXT3	TIM2_CH2	Internal signal from on-chip timers	0011
EXT4	TIM3_TRGO	Internal signal from on-chip timers	0100
EXT5	TIM4_CH4	Internal signal from on-chip timers	0101
EXT6	EXTI line 11	External pin	0110
EXT7	TIM8_TRGO	Internal signal from on-chip timers	0111
EXT8	TIM8_TRGO2	Internal signal from on-chip timers	1000
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
EXT12	TIM4_TRGO	Internal signal from on-chip timers	1100
EXT13	TIM6_TRGO	Internal signal from on-chip timers	1101
EXT14	TIM15_TRGO	Internal signal from on-chip timers	1110
EXT15	TIM3_CH4	Internal signal from on-chip timers	1111

**Table 119. ADC1 - External trigger for injected channels**

Name	Source	Type	JEXTSEL[3..0]
JEXT0	TIM1_TRGO	Internal signal from on-chip timers	0000
JEXT1	TIM1_CH4	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	TIM3_CH4	Internal signal from on-chip timers	0100
JEXT5	TIM4_TRGO	Internal signal from on-chip timers	0101
JEXT6	EXTI line 15	External pin	0110
JEXT7	TIM8_CH4	Internal signal from on-chip timers	0111
JEXT8	TIM1_TRGO2	Internal signal from on-chip timers	1000
JEXT9	TIM8_TRGO	Internal signal from on-chip timers	1001
JEXT10	TIM8_TRGO2	Internal signal from on-chip timers	1010

Table 119. ADC1 - External trigger for injected channels

Name	Source	Type	JEXTSEL[3..0]
JEXT11	TIM3_CH3	Internal signal from on-chip timers	1011
JEXT12	TIM3_TRGO	Internal signal from on-chip timers	1100
JEXT13	TIM3_CH1	Internal signal from on-chip timers	1101
JEXT14	TIM6_TRGO	Internal signal from on-chip timers	1110
JEXT15	TIM15_TRGO	Internal signal from on-chip timers	1111

## 21.4.19 Injected channel management

### Triggered injection mode

To use triggered injection, the JAUTO bit in the ADCx\_CFGR register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADCx\_CR register.
2. If an external injected trigger occurs, or if the JADSTART bit in the ADCx\_CR register is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches are launched (all the injected channels are converted once).
3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence.

*Figure 95* shows the corresponding timing diagram.

*Note:* When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence. For instance, if the sequence length is 28 ADC clock cycles (that is two conversions with a sampling time of 1.5 clock periods), the minimum interval between triggers must be 29 ADC clock cycles.

### Auto-injection mode

If the JAUTO bit in the ADCx\_CFGR register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADCx\_SQRy and ADCx\_JSQR registers.

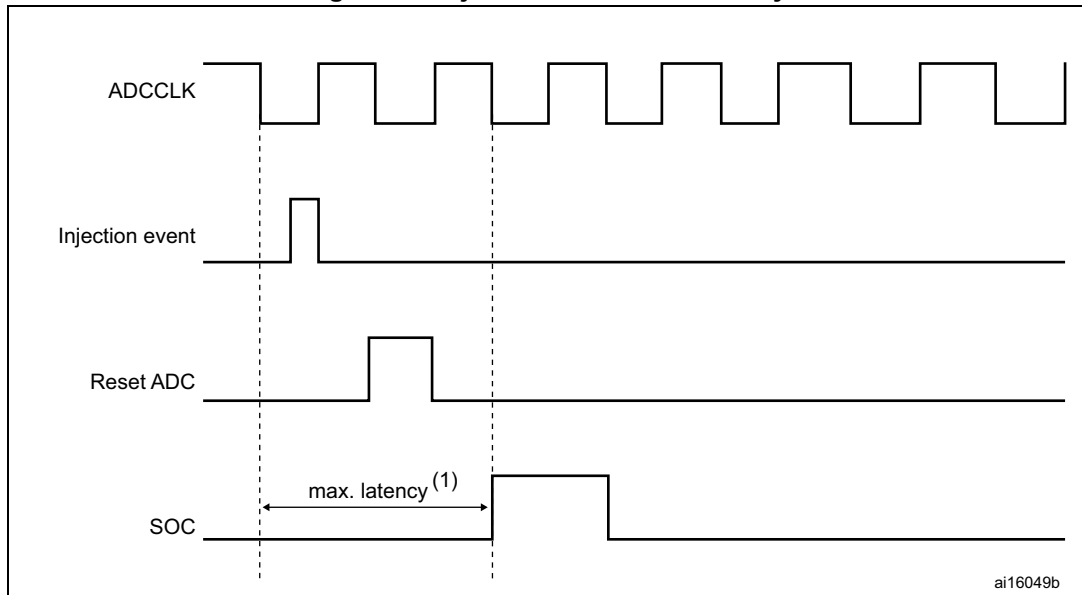
In this mode, the ADSTART bit in the ADCx\_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

*Note:* It is not possible to use both the auto-injected and discontinuous modes simultaneously. When the DMA is used for exporting regular sequencer's data in JAUTO mode, it is necessary to program it in circular mode (CIRC bit set in DMA\_CCRx register). If the CIRC bit is reset (single-shot mode), the JAUTO sequence will be stopped upon DMA Transfer Complete event.

Figure 95. Injected conversion latency



1. The maximum latency value can be found in the electrical characteristics of the STM32L4Rxxx and STM32L4Sxxx datasheet.

### 21.4.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

#### Regular group mode

This mode is enabled by setting the DISCEN bit in the ADCx\_CFGR register.

It is used to convert a short sequence (subgroup) of  $n$  conversions ( $n \leq 8$ ) that is part of the sequence of conversions selected in the ADCx\_SQRy registers. The value of  $n$  is specified by writing to the DISCNUM[2:0] bits in the ADCx\_CFGR register.

When an external trigger occurs, it starts the next  $n$  conversions selected in the ADCx\_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 ADCx\_SQR1 register.

Example:

- DISCEN=1,  $n=3$ , channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - 2nd trigger: channels converted are 6, 7, 8 (an EOC event is generated at each conversion).
  - 3rd trigger: channels converted are 9, 10, 11 (an EOC event is generated at each conversion) and an EOS event is generated after the conversion of channel 11.
  - 4th trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).

- ...
- DISCEN=0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10,11
  - 1st trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 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 will relaunch the complete sequence.

*Note:* When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

*When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the 4th trigger reconverts the channels 1, 2 and 3 in the 1st subgroup.*

*It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN=1, CONT=1), the ADC behaves as if continuous mode was disabled.*

### Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADCx\_CFGR register. It converts the sequence selected in the ADCx\_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels where 'n' is fixed to 1.

When an external trigger occurs, it starts the next channel conversions selected in the ADCx\_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADCx\_JSQR register.

Example:

- JDISCEN=1, channels to be converted = 1, 2, 3
  - 1st trigger: channel 1 converted (a JEEOC event is generated)
  - 2nd trigger: channel 2 converted (a JEEOC event is generated)
  - 3rd trigger: channel 3 converted and a JEEOC event + a JEEOC event are generated
  - ...

*Note:* When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel 1.

*It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.*

### 21.4.21 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 ADCx\_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 ADCx\_JSQR register)
- Definition of the injected sequence (bits JSQx[4:0] and JL[1:0] in ADCx\_JSQR register)

All the parameters of the context are defined into a single register ADCx\_JSQR and this register implements a queue of 2 buffers, allowing the bufferization of up to 2 sets of parameters:

- The JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written into the JSQR register is stored into the Queue of context.
- At the beginning, the Queue is empty and the first write access into the JSQR register immediately changes the context and the ADC is ready to receive injected triggers.
- Once an injected sequence is complete, the Queue is consumed and the context changes according to the next JSQR parameters stored in the Queue. This new context is applied for the next injected sequence of conversions.
- A Queue overflow occurs when writing into register JSQR while the Queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two possible behaviors are possible when the Queue becomes empty, depending on the value of the control bit JQM of register ADCx\_CFGR:
  - If JQM=0, the Queue is empty just after enabling the ADC, but then it can never be empty during run operations: the Queue always maintains the last active context and any further valid start of injected sequence will be served according to the last active context.
  - If JQM=1, the Queue can be empty after the end of an injected sequence or if the Queue is flushed. When this occurs, there is no more context in the queue and 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.

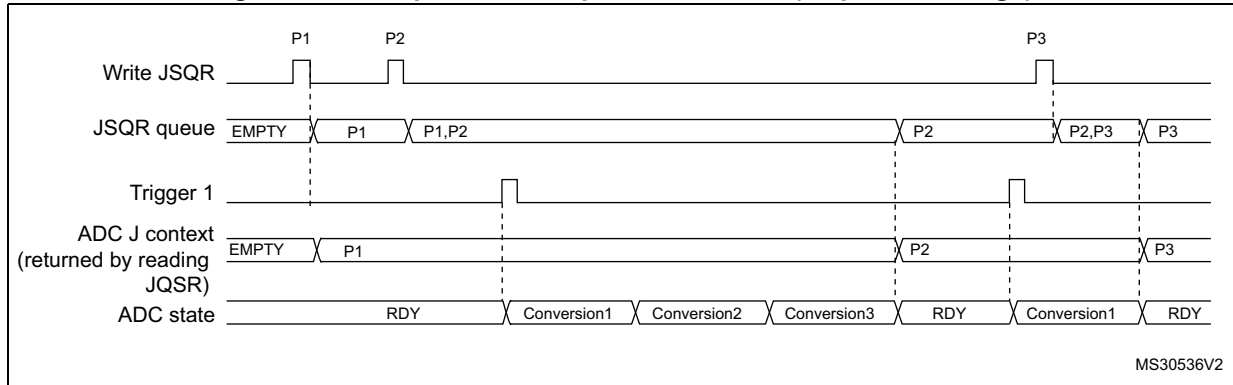
*Note:* When configured in discontinuous mode (bit JDISCEN=1), only the last trigger of the injected sequence changes the context and consumes the Queue. The 1<sup>st</sup> trigger only consumes the queue but others are still valid triggers as shown by the discontinuous mode example below (length = 3 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, disc. 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**

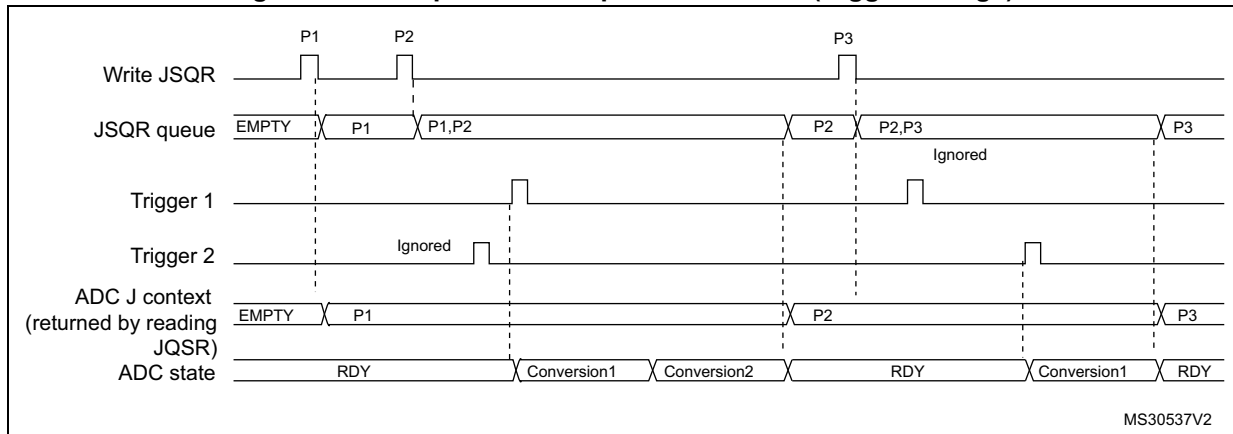
The [Figure 96](#) and [Figure 97](#) show the behavior of the context Queue when changing the sequence or the triggers.

**Figure 96. Example of JSQR queue of context (sequence change)**



- 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 97. Example of JSQR queue of context (trigger change)**

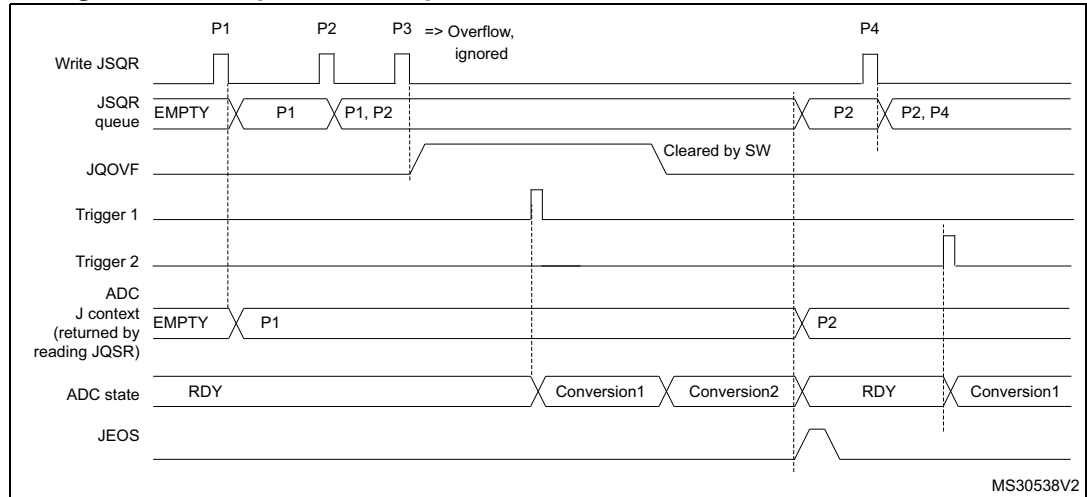


- Parameters:
  - P1: sequence of 2 conversions, hardware trigger 1
  - P2: sequence of 1 conversion, hardware trigger 2
  - P3: sequence of 4 conversions, hardware trigger 1

**Queue of context: Behavior when a queue overflow occurs**

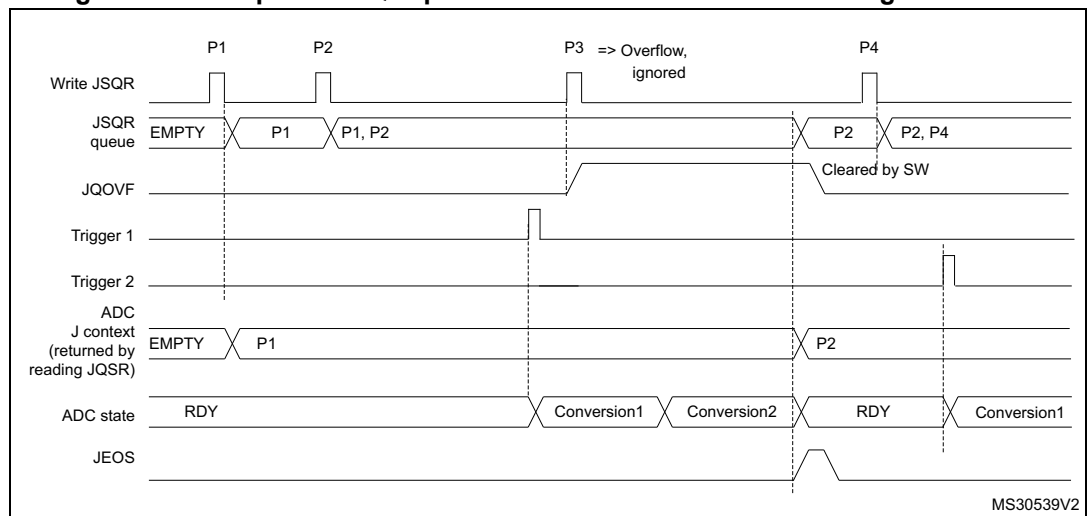
The [Figure 98](#) and [Figure 99](#) show the behavior of the context Queue if an overflow occurs before or during a conversion.

**Figure 98. Example of JSQR queue of context with overflow before conversion**



- Parameters:
  - P1: sequence of 2 conversions, hardware trigger 1
  - P2: sequence of 1 conversion, hardware trigger 2
  - P3: sequence of 3 conversions, hardware trigger 1
  - P4: sequence of 4 conversions, hardware trigger 1

**Figure 99. Example of JSQR queue of context with overflow during conversion**



- Parameters:
  - P1: sequence of 2 conversions, hardware trigger 1
  - P2: sequence of 1 conversion, hardware trigger 2
  - P3: sequence of 3 conversions, hardware trigger 1
  - P4: sequence of 4 conversions, hardware trigger 1



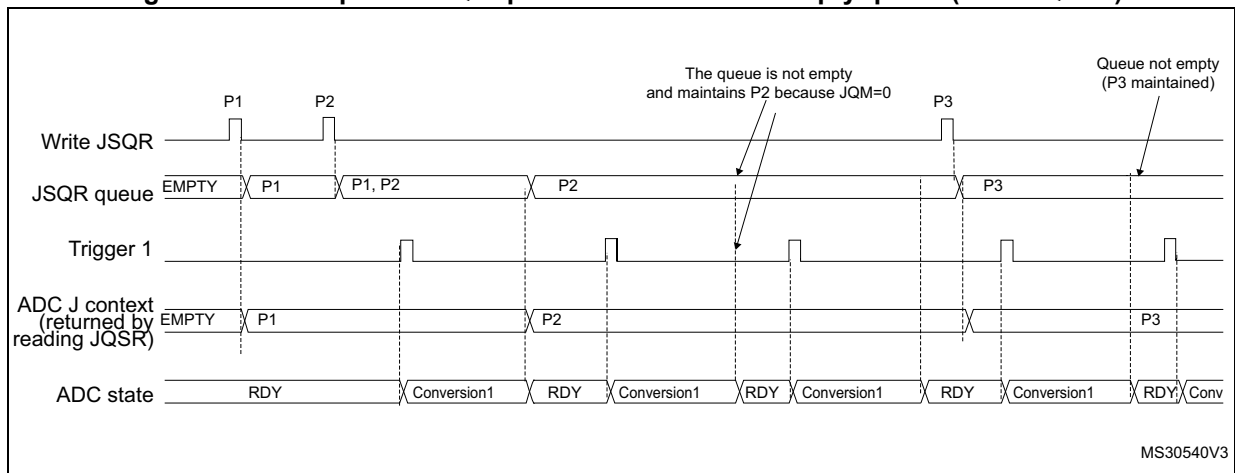
It is recommended to manage the queue overflows as described below:

- After each P context write into JSQR register, flag JQOVF shows if the write has been ignored or not (an interrupt can be generated).
- Avoid Queue overflows by writing the third context (P3) only once the flag JEOS of the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

**Queue of context: Behavior when the queue becomes empty**

Figure 100 and Figure 101 show the behavior of the context Queue when the Queue becomes empty in both cases JQM=0 or 1.

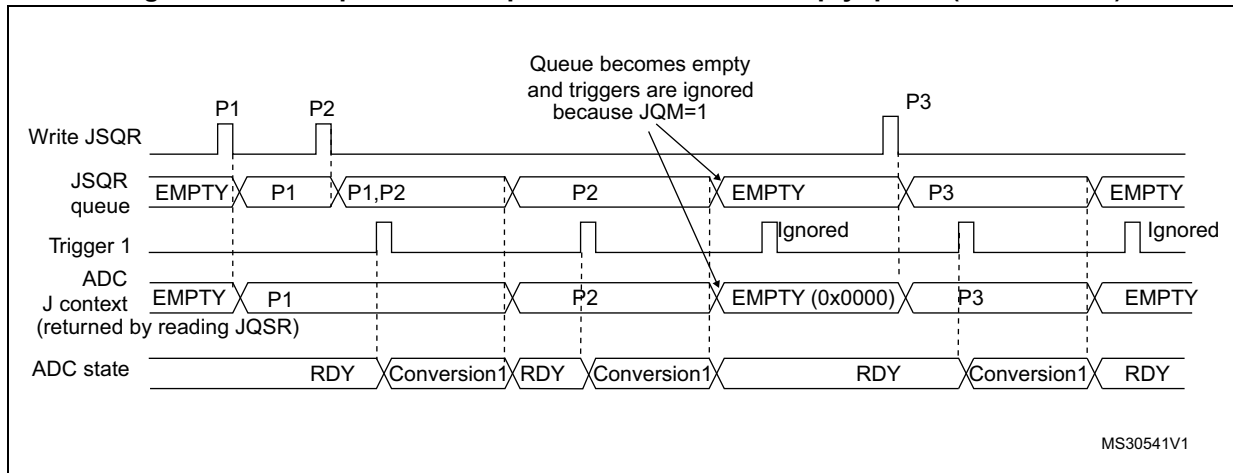
**Figure 100. Example of JSQR queue of context with empty queue (case JQM=0)**



- Parameters:
  - P1: sequence of 1 conversion, hardware trigger 1
  - P2: sequence of 1 conversion, hardware trigger 1
  - P3: sequence of 1 conversion, hardware trigger 1

**Note:** When writing P3, the context changes immediately. However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately.

Figure 101. Example of JSQR queue of context with empty queue (case JQM=1)

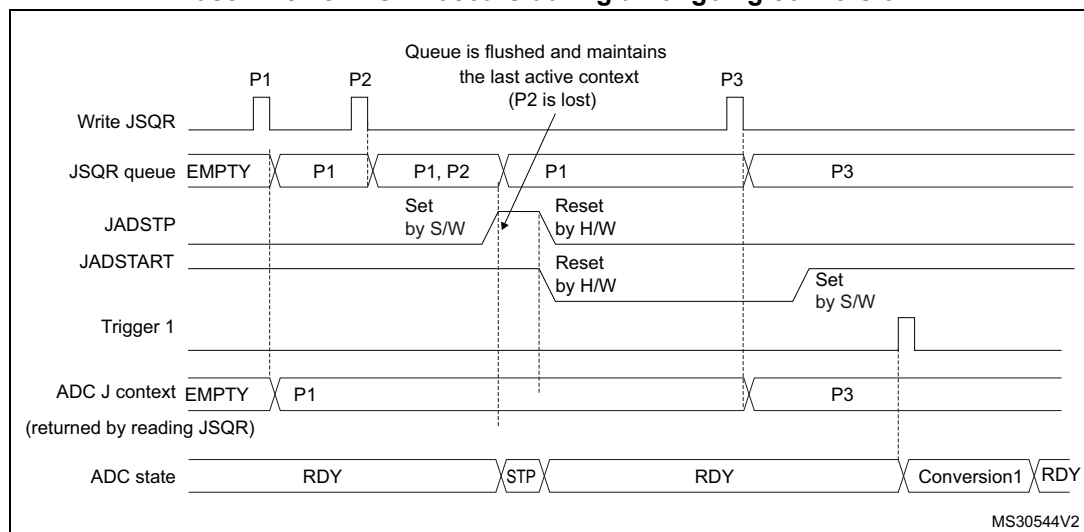


- Parameters:
  - P1: sequence of 1 conversion, hardware trigger 1
  - P2: sequence of 1 conversion, hardware trigger 1
  - P3: sequence of 1 conversion, hardware trigger 1

### Flushing the queue of context

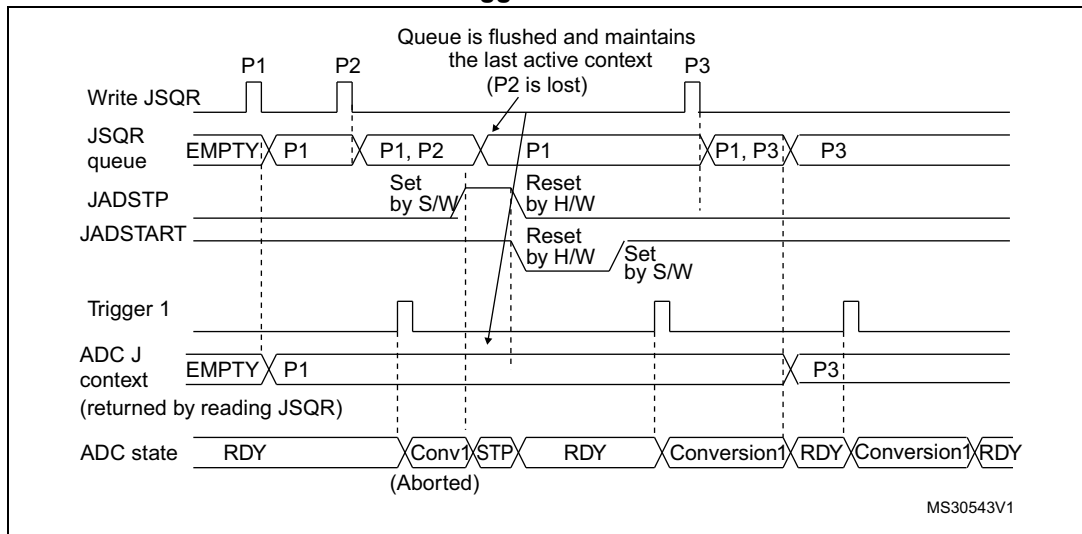
The figures below show the behavior of the context Queue in various situations when the queue is flushed.

Figure 102. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion.



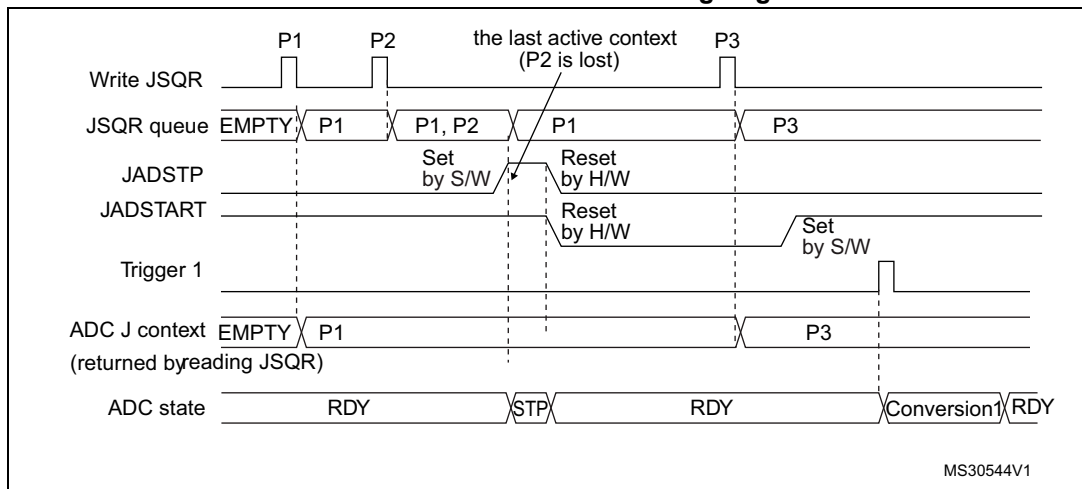
- Parameters:
  - P1: sequence of 1 conversion, hardware trigger 1
  - P2: sequence of 1 conversion, hardware trigger 1
  - P3: sequence of 1 conversion, hardware trigger 1

**Figure 103. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0).  
Case when JADSTP occurs during an ongoing conversion and a new trigger occurs.**



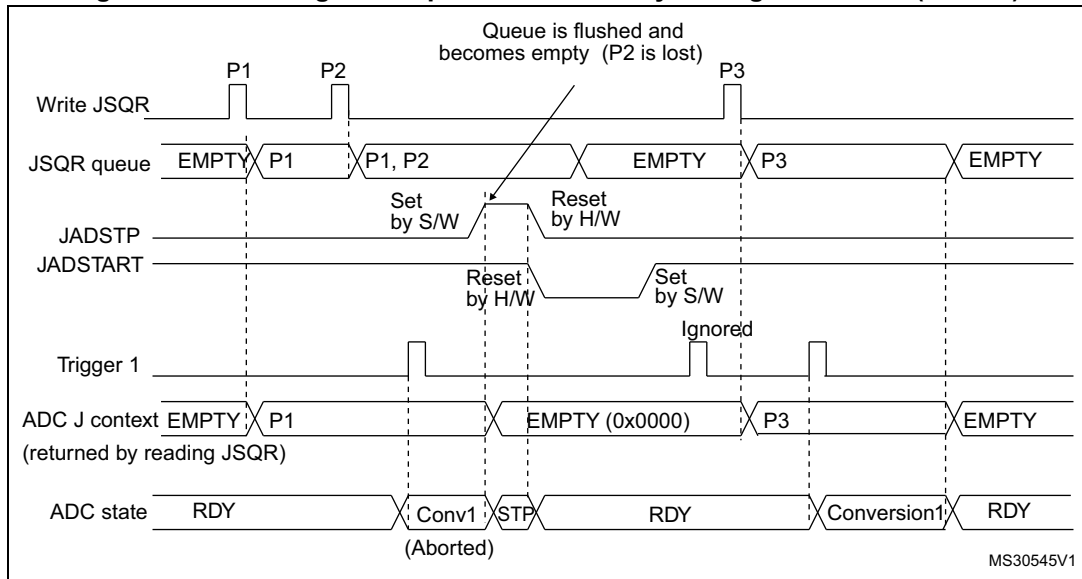
- Parameters:  
 P1: sequence of 1 conversion, hardware trigger 1  
 P2: sequence of 1 conversion, hardware trigger 1  
 P3: sequence of 1 conversion, hardware trigger 1

**Figure 104. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0).  
Case when JADSTP occurs outside an ongoing conversion**



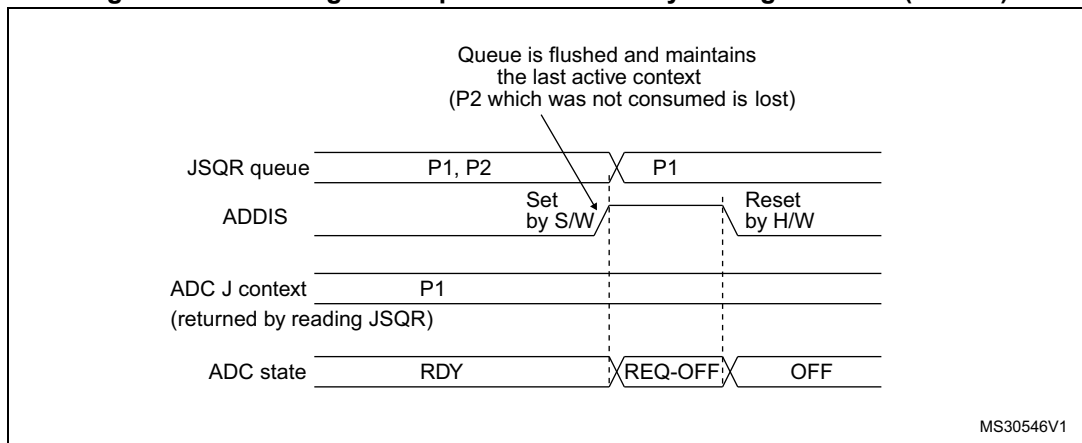
- Parameters:  
 P1: sequence of 1 conversion, hardware trigger 1  
 P2: sequence of 1 conversion, hardware trigger 1  
 P3: sequence of 1 conversion, hardware trigger 1

**Figure 105. Flushing JSQR queue of context by setting JADSTP=1 (JQM=1)**



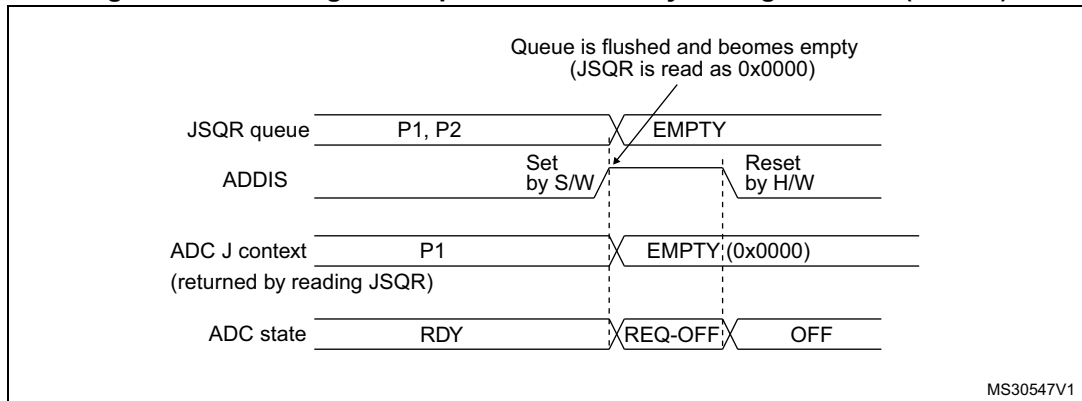
- Parameters:
  - P1: sequence of 1 conversion, hardware trigger 1
  - P2: sequence of 1 conversion, hardware trigger 1
  - P3: sequence of 1 conversion, hardware trigger 1

**Figure 106. Flushing JSQR queue of context by setting ADDIS=1 (JQM=0)**



- Parameters:
  - P1: sequence of 1 conversion, hardware trigger 1
  - P2: sequence of 1 conversion, hardware trigger 1
  - P3: sequence of 1 conversion, hardware trigger 1

**Figure 107. 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 not equal to 0 (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 ADCx\_CFGR register.

**21.4.22 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 112](#), [Figure 113](#), [Figure 114](#) and [Figure 115](#) 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 120](#).

Table 120.  $T_{SAR}$  timings depending on resolution

RES (bits)	$T_{SAR}$ (ADC clock cycles)	$T_{SAR}$ (ns) at $F_{ADC}=80$ MHz	$T_{CONV}$ (ADC clock cycles) (with Sampling Time= 2.5 ADC clock cycles)	$T_{CONV}$ (ns) at $F_{ADC}=80$ MHz
12	12.5 ADC clock cycles	156.25 ns	15 ADC clock cycles	187.5 ns
10	10.5 ADC clock cycles	131.25 ns	13 ADC clock cycles	162.5 ns
8	8.5 ADC clock cycles	106.25 ns	11 ADC clock cycles	137.5 ns
6	6.5 ADC clock cycles	81.25 ns	9 ADC clock cycles	112.5 ns

### 21.4.23 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 ADCx\_DR register. An interrupt can be generated if bit EOCIE is set. EOC flag is cleared by the software either by writing 1 to it or by reading ADCx\_DR.

The ADC sets the JEOC flag as soon as a new injected conversion data is available in one of the ADCx\_JDRy register. An interrupt can be generated if bit JEOCIE is set. JEOC flag is cleared by the software either by writing 1 to it or by reading the corresponding ADCx\_JDRy register.

The ADC also notifies the end of Sampling phase by setting the status bit EOSMP (for regular conversions only). EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if bit EOSMPIE is set.

### 21.4.24 End of conversion sequence (EOS, JEOS)

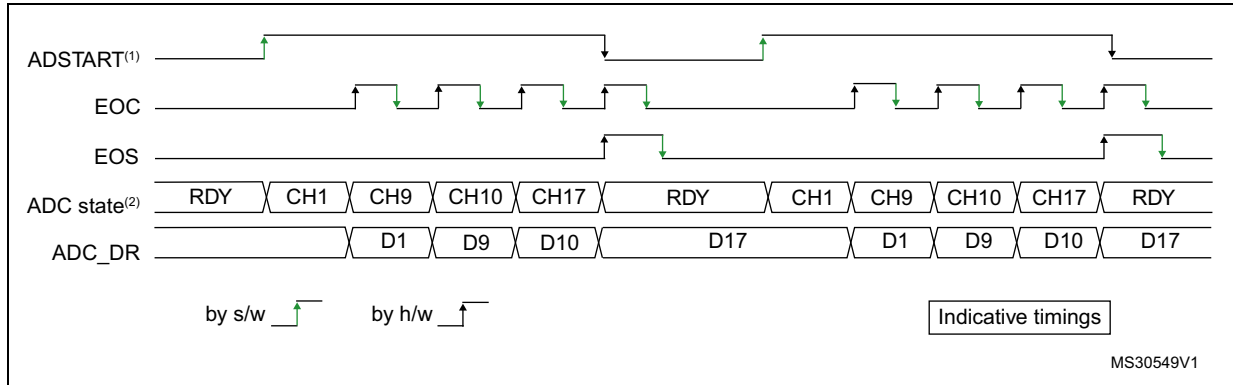
The ADC notifies the application for each end of regular sequence (EOS) and for each end of injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the last data of the regular conversion sequence is available in the ADCx\_DR register. An interrupt can be generated if bit EOSIE is set. EOS flag is cleared by the software either by writing 1 to it.

The ADC sets the JEOS flag as soon as the last data of the injected conversion sequence is complete. An interrupt can be generated if bit JEOSIE is set. JEOS flag is cleared by the software either by writing 1 to it.

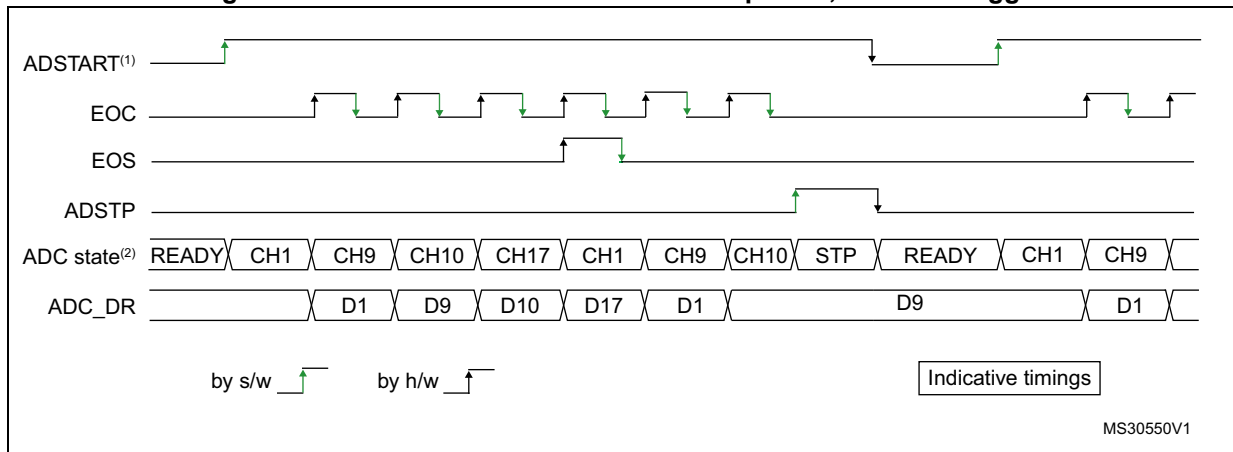
**21.4.25 Timing diagrams example (single/continuous modes, hardware/software triggers)**

**Figure 108. Single conversions of a sequence, software trigger**



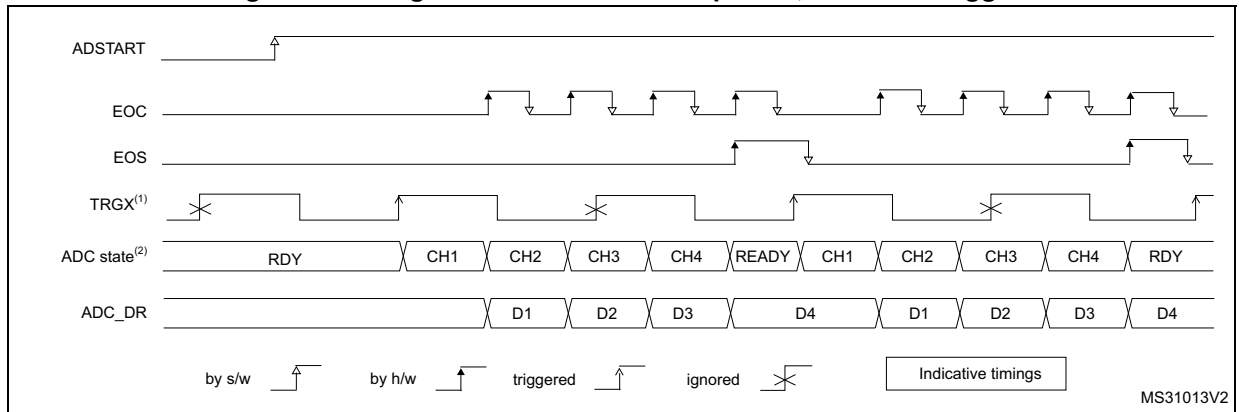
1. EXTEN=0x0, CONT=0
2. Channels selected = 1,9, 10, 17; AUTDLY=0.

**Figure 109. Continuous conversion of a sequence, software trigger**



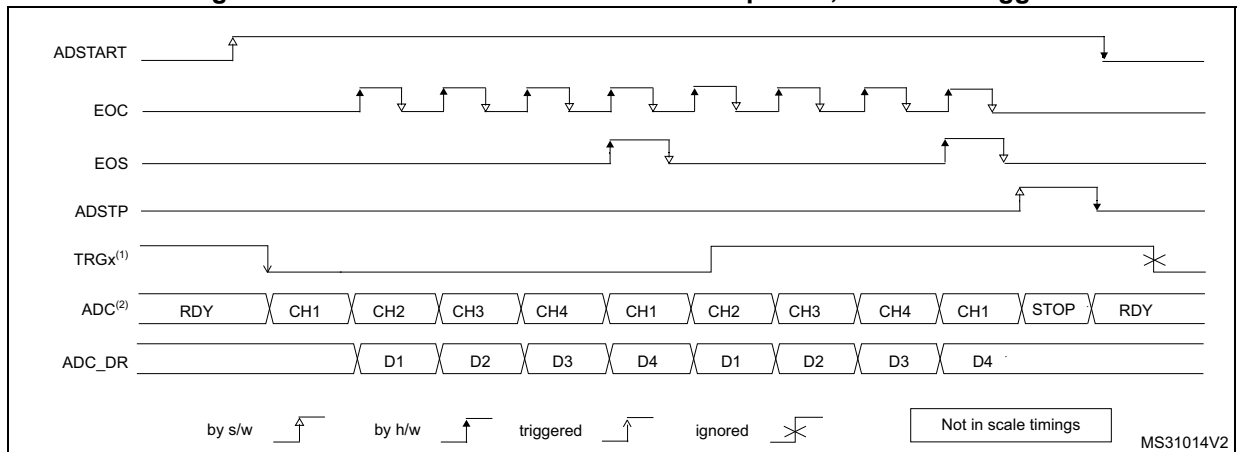
1. EXTEN=0x0, CONT=1
2. Channels selected = 1,9, 10, 17; AUTDLY=0.

Figure 110. Single conversions of a sequence, hardware trigger



1. TRGX (over-frequency) is selected as trigger source, EXTEN = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY=0.

Figure 111. Continuous conversions of a sequence, hardware trigger



1. TRGX is selected as trigger source, EXTEN = 10, CONT = 1
2. Channels selected = 1, 2, 3, 4; AUTDLY=0.



### 21.4.26 Data management

#### Data register, data alignment and offset (ADCx\_DR, OFFSETy, OFFSETy\_CH, ALIGN)

##### Data and alignment

At the end of each regular conversion channel (when EOC event occurs), the result of the converted data is stored into the ADCx\_DR data register which is 16 bits wide.

At the end of each injected conversion channel (when JEOC event occurs), the result of the converted data is stored into the corresponding ADCx\_JDRy data register which is 16 bits wide.

The ALIGN bit in the ADCx\_CFGR register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in [Figure 112](#), [Figure 113](#), [Figure 114](#) and [Figure 115](#).

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 114](#) and [Figure 115](#).

*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 ADCx\_OFRRy register. The channel to which the offset will be applied is programmed into the bits OFFSETy\_CH[4:0] of ADCx\_OFRRy 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 ADCx\_OFRRy register is ignored (considered as reset).

[Table 123](#) describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

**Table 121. 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 121. Offset computation versus data resolution (continued)**

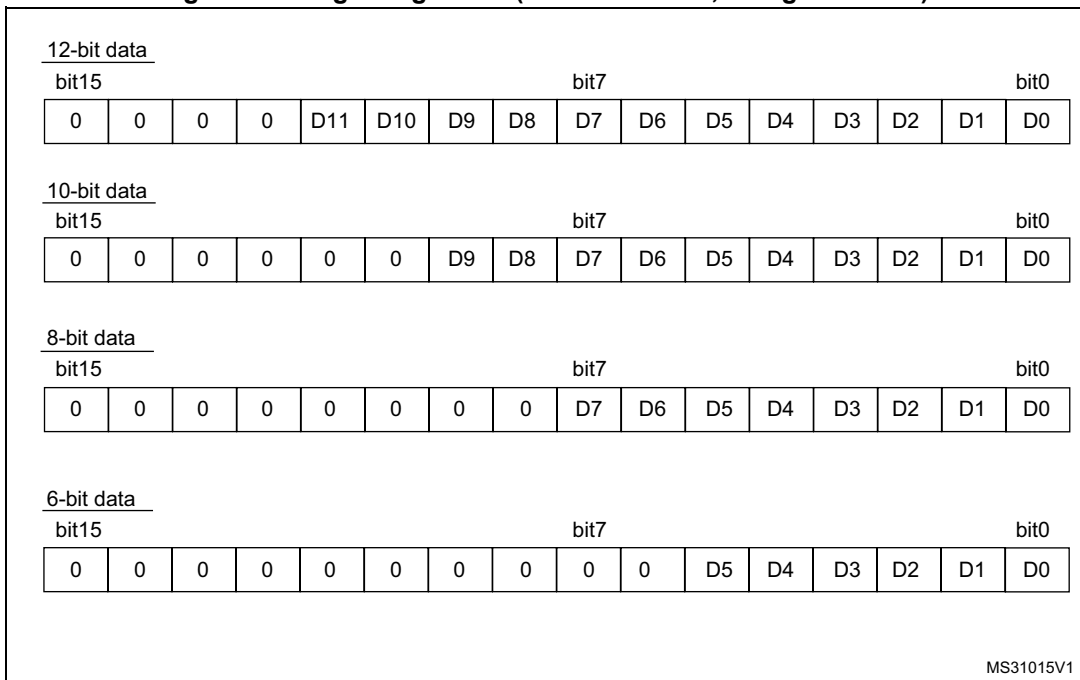
Resolution (bits RES[1:0])	Subtraction between raw converted data and offset		Result	Comments
	Raw converted Data, left aligned	Offset		
10: 8-bit	DATA[11:4],0000	OFFSET[11:0]	Signed 8-bit data	The user must configure OFFSET[3:0] to "0000"
11: 6-bit	DATA[11:6],000000	OFFSET[11:0]	Signed 6-bit data	The user must configure OFFSET[5:0] to "000000"

When reading data from ADCx\_DR (regular channel) or from ADCx\_JDRy (injected channel, y=1,2,3,4) corresponding to the channel "i":

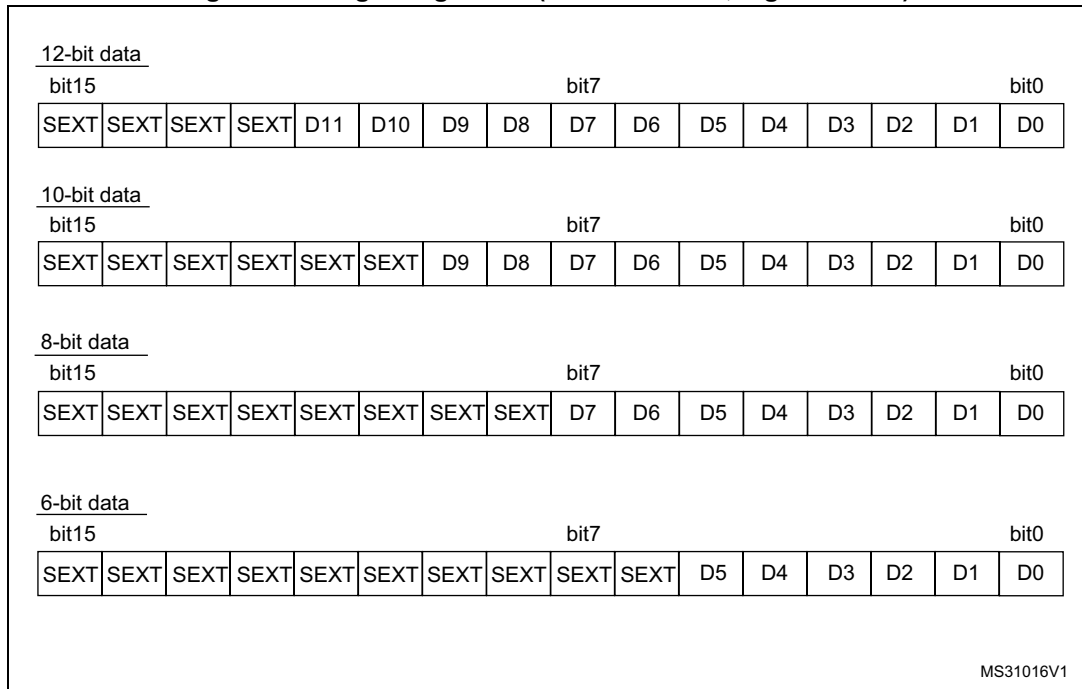
- If one of the offsets is enabled (bit OFFSETy\_EN=1) for the corresponding channel, the read data is signed.
- If none of the four offsets is enabled for this channel, the read data is not signed.

Figure 112, Figure 113, Figure 114 and Figure 115 show alignments for signed and unsigned data.

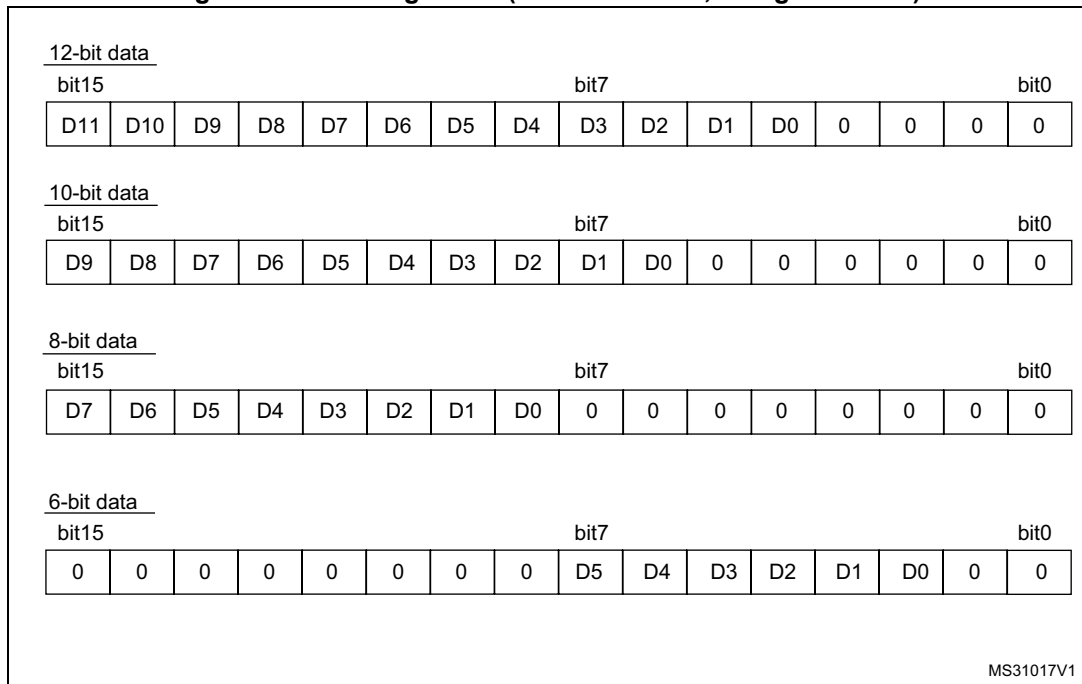
**Figure 112. Right alignment (offset disabled, unsigned value)**



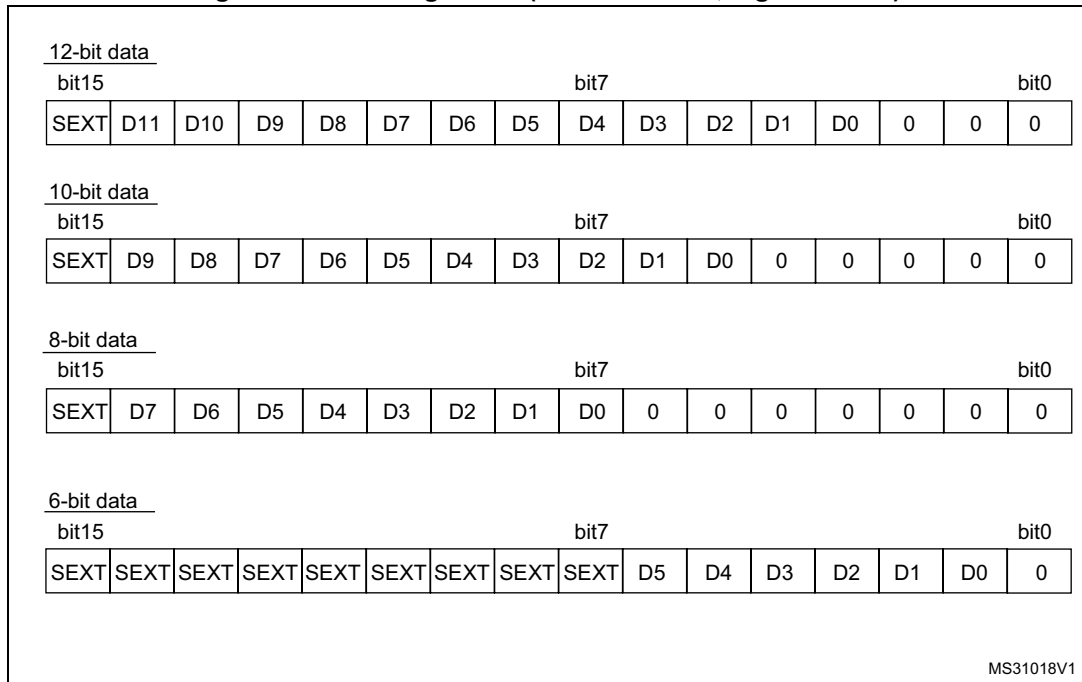
**Figure 113. Right alignment (offset enabled, signed value)**



**Figure 114. Left alignment (offset disabled, unsigned value)**



**Figure 115. Left alignment (offset enabled, signed value)**



**ADC overrun (OVR, OVRMOD)**

The overrun flag (OVR) notifies of a buffer overrun event, when the regular converted data was not read (by the CPU or the DMA) before new converted data became available.

The OVR flag is set if the EOC flag is still 1 at the time when a new conversion completes. An interrupt can be generated if bit OVRIE=1.

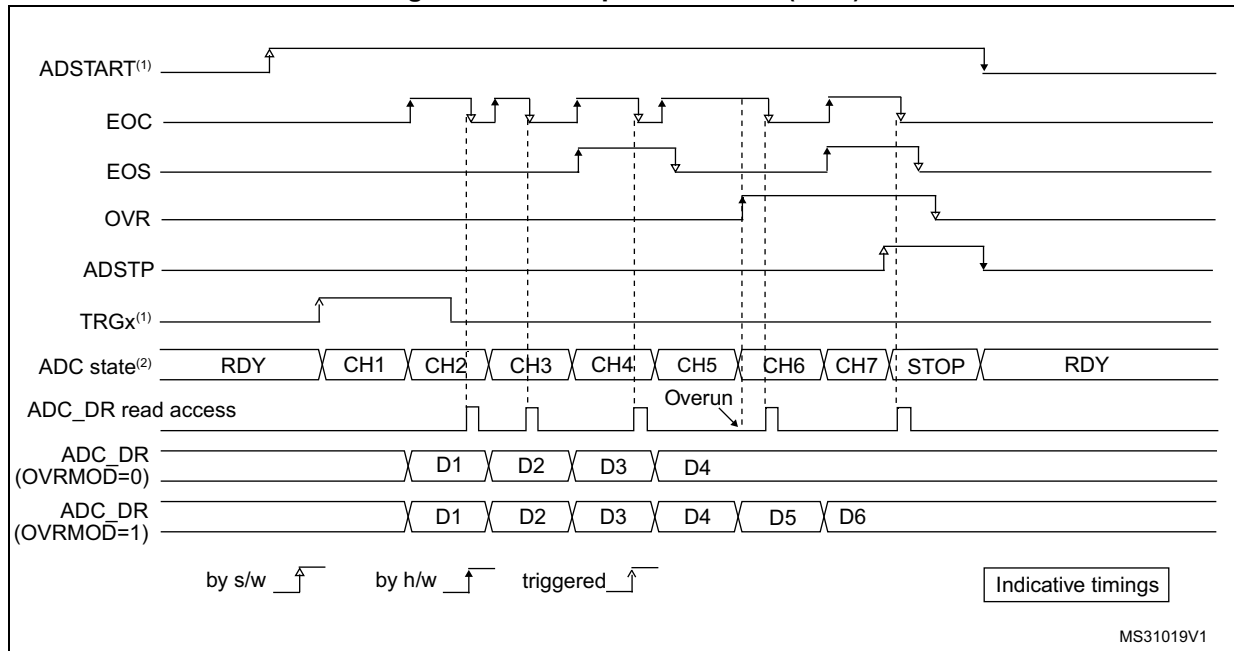
When an overrun condition occurs, the ADC is still operating and can continue to convert unless the software decides to stop and reset the sequence by setting bit ADSTP=1.

OVR flag is cleared by software by writing 1 to it.

It is possible to configure if data is preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD=0: The overrun event preserves the data register from being overrun: the old data is maintained and the new conversion is discarded and lost. If OVR remains at 1, any further conversions will occur but the result data will be also discarded.
- OVRMOD=1: The data register is overwritten with the last conversion result and the previous unread data is lost. If OVR remains at 1, any further conversions will operate normally and the ADCx\_DR register will always contain the latest converted data.

Figure 116. Example of overrun (OVR)



Note: There is no overrun detection on the injected channels since there is a dedicated data register for each of the four injected channels.

### Managing a sequence of conversion without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADCx\_DR register can be read. OVRMOD should be configured to 0 to manage overrun events as an error.

### Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and OVR flag should be ignored by the software. An overrun event will not prevent the ADC from continuing to convert and the ADCx\_DR register will always contain the latest conversion.

### Managing conversions using the DMA

Since converted channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one channel. This avoids the loss of the data already stored in the ADCx\_DR register.

When the DMA mode is enabled (DMAEN bit set to 1 in the ADCx\_CFGR register, a DMA request is generated after each conversion of a channel. This allows the transfer of the converted data from the ADCx\_DR register to the destination location selected by the software.

Despite this, if an overrun occurs ( $OVR=1$ ) because the DMA could not serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of  $OVRMOD$  bit, the data is either preserved or overwritten (refer to [Section : ADC overrun \(OVR, OVRMOD\)](#)).

The DMA transfer requests are blocked until the software clears the  $OVR$  bit.

Two different DMA modes are proposed depending on the application use and are configured with bit  $DMACFG$  of the  $ADCx\_CFGR$  register:

- DMA one shot mode ( $DMACFG=0$ ).  
This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode ( $DMACFG=1$ )  
This mode is suitable when programming the DMA in circular mode.

#### **DMA one shot mode (DMACFG=0)**

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when  $DMA\_EOT$  interrupt occurs - refer to DMA paragraph) even if a conversion has been started again.

When the DMA transfer is complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- Any ongoing conversion is aborted with partial result discarded.
- No new DMA request is issued to the DMA controller. This avoids generating an overrun error if there are still conversions which are started.
- Scan sequence is stopped and reset.
- The DMA is stopped.

#### **DMA circular mode (DMACFG=1)**

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available in the data register, even if the DMA has reached the last DMA transfer. This allows configuring the DMA in circular mode to handle a continuous analog input data stream.

### **21.4.27 Managing conversions using the DFSDM**

The ADC conversion results can be transferred directly to the Digital filter for sigma delta modulators (DFSDM).

In this case, the  $DFSDMCFG$  bit must be set to 1 and  $DMAEN$  bit must be cleared to 0.

The ADC transfers all the 16 bits of the regular data register data to the DFSDM and resets the  $EOC$  flag once the transfer is complete.

The data format must be 16-bit signed:

$ADCx\_DR[15:12]$  = sign extended  
 $ADCx\_DR[11]$  = sign  
 $ADCx\_DR[11:0]$  = data

To obtain 16-bit signed format in 12-bit ADC mode, the software needs to configure the OFFSETy[11:0] to 0x800 after having set OFFSETy\_EN to 1.

Only right aligned data format is available for the DFSDM interface (see [Figure 113: Right alignment \(offset enabled, signed value\)](#)).

## 21.4.28 Dynamic low-power features

### Auto-delayed conversion mode (AUTDLY)

The ADC implements an auto-delayed conversion mode controlled by the AUTDLY configuration bit. Auto-delayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there would be risk of encountering an ADC overrun.

When AUTDLY=1, a new conversion can start only if all the previous data of the same group has been treated:

- For a regular conversion: once the ADCx\_DR register has been read or if the EOC bit has been cleared (see [Figure 117](#)).
- For an injected conversion: when the JEOS bit has been cleared (see [Figure 118](#)).

This is a way to automatically adapt the speed of the ADC to the speed of the system which will read the data.

The delay is inserted after each regular conversion (whatever DISCEN=0 or 1) and after each sequence of injected conversions (whatever JDISCEN=0 or 1).

*Note:* There is no delay inserted between each conversions of the injected sequence, except after the last one.

During a conversion, a hardware trigger event (for the same group of conversions) occurring during this delay is ignored.

*Note:* This is not true for software triggers where it remains possible during this delay to set the bits ADSTART or JADSTART to 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 118](#)).
- 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 120](#)).

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 121](#)).

To stop a conversion in continuous auto-injection mode combined with autodelay mode (JAUTO=1, CONT=1 and AUTDLY=1), follow the following procedure:

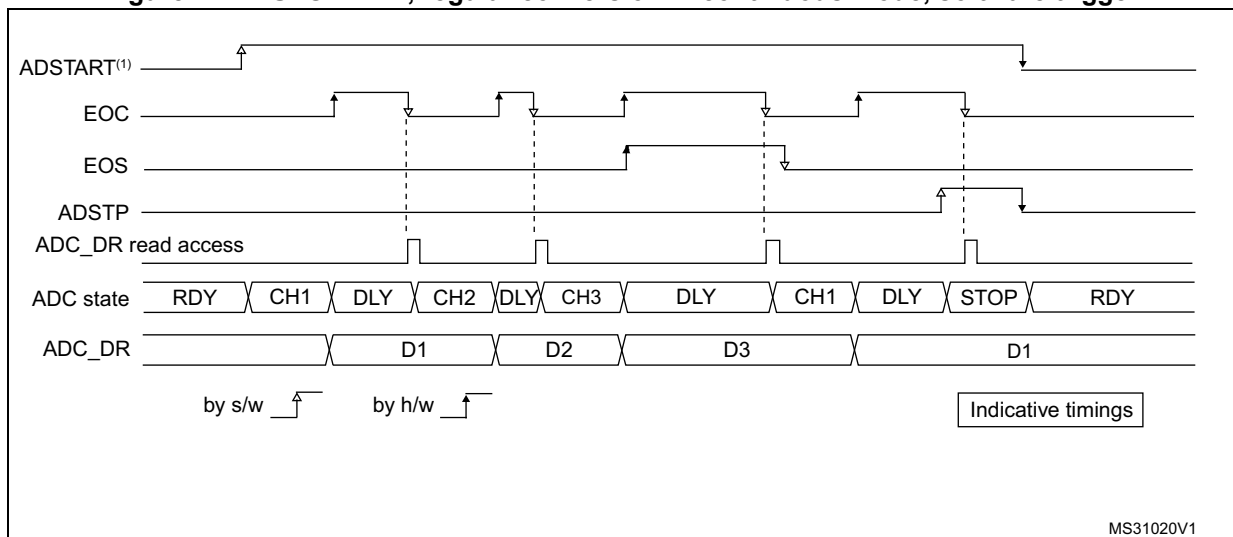
1. Wait until JEOS=1 (no more conversions are restarted)
2. Clear JEOS,
3. Set ADSTP=1
4. Read the regular data.

If this procedure is not respected, a new regular sequence can 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 or the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

In AUTDLY mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

**Figure 117. AUTODLY=1, regular conversion in continuous mode, software trigger**

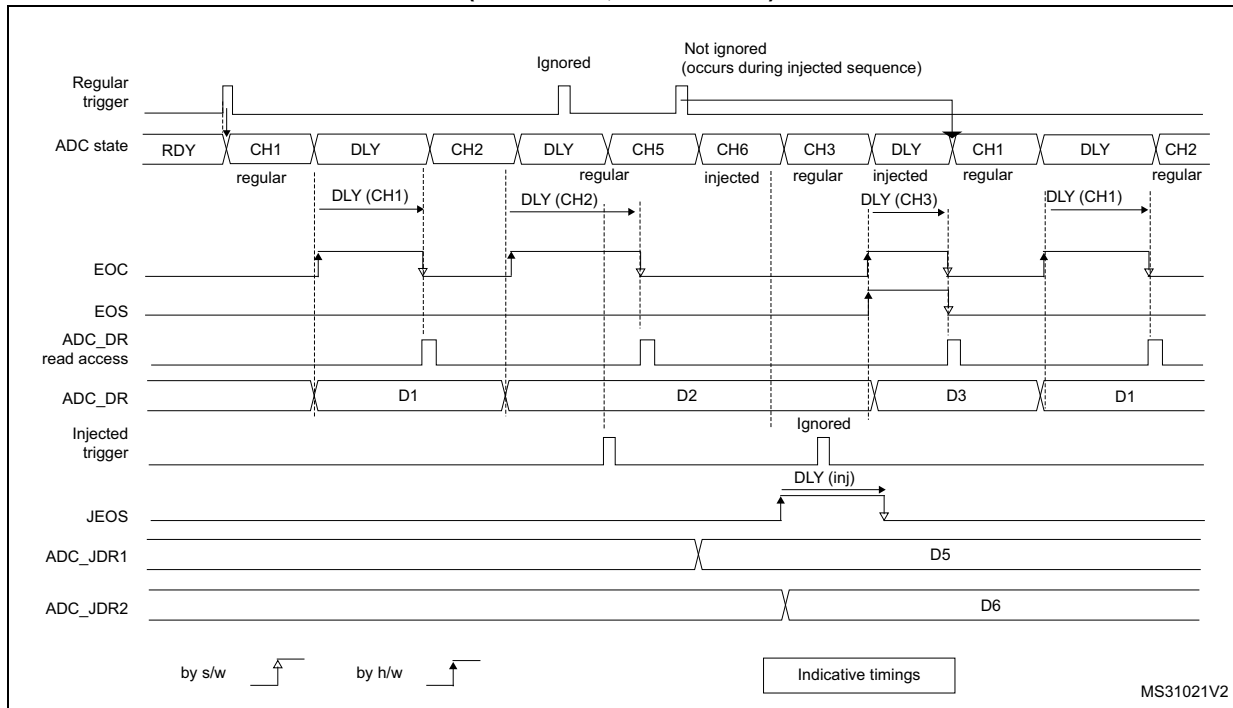


MS31020V1

1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, CHANNELS = 1,2,3
3. Injected configuration DISABLED

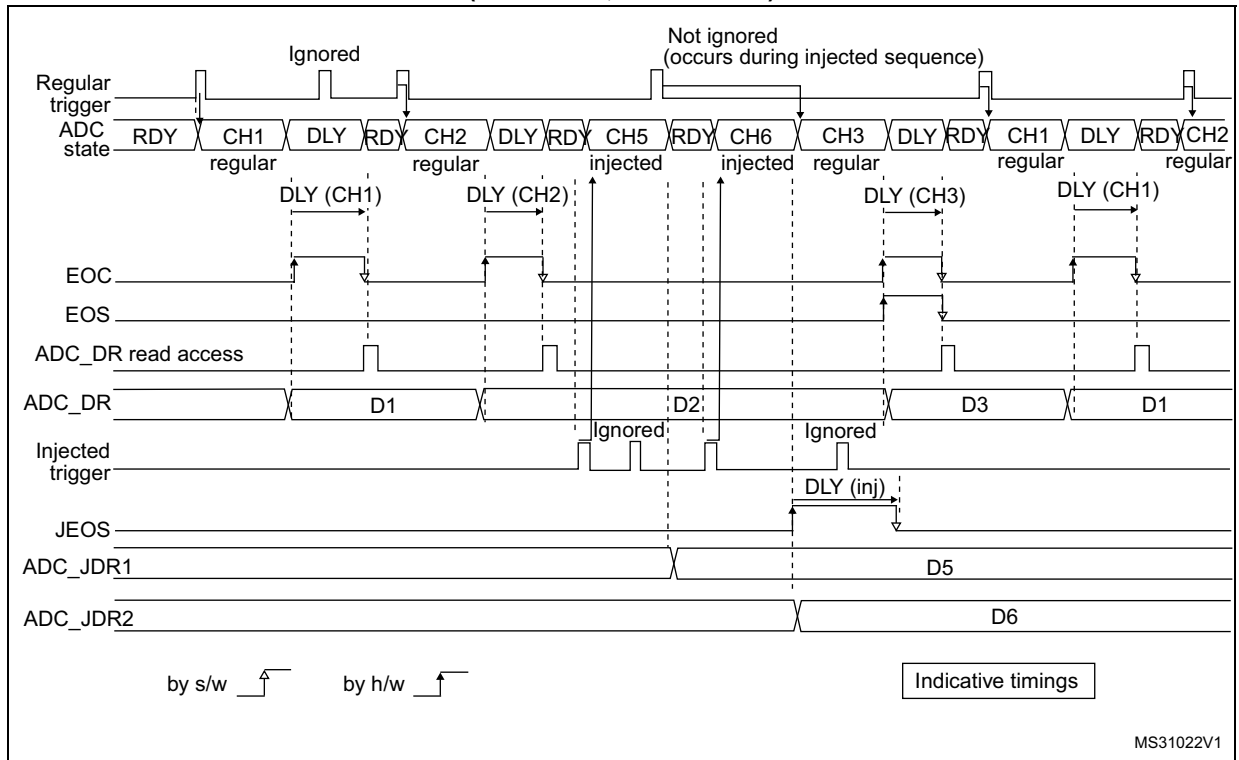


**Figure 118. AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=0; JDISCEN=0)**



1. AUTODLY=1
2. Regular configuration: EXTEN=0x1 (HW trigger), CONT=0, DISCEN=0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=0, CHANNELS = 5,6

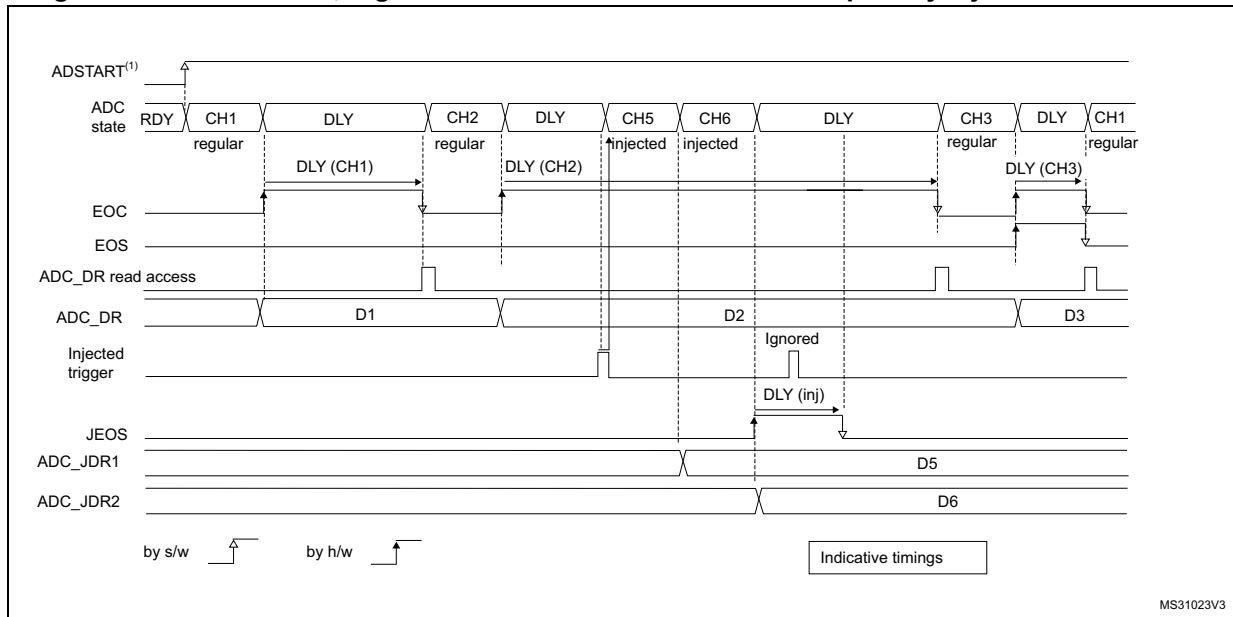
**Figure 119. AUTDLY=1, regular HW conversions interrupted by injected conversions (DISCEN=1, JDISCEN=1)**



MS31022V1

1. AUTDLY=1
2. Regular configuration: EXTEN=0x1 (HW trigger), CONT=0, DISCEN=1, DISCNUM=1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=1, CHANNELS = 5,6

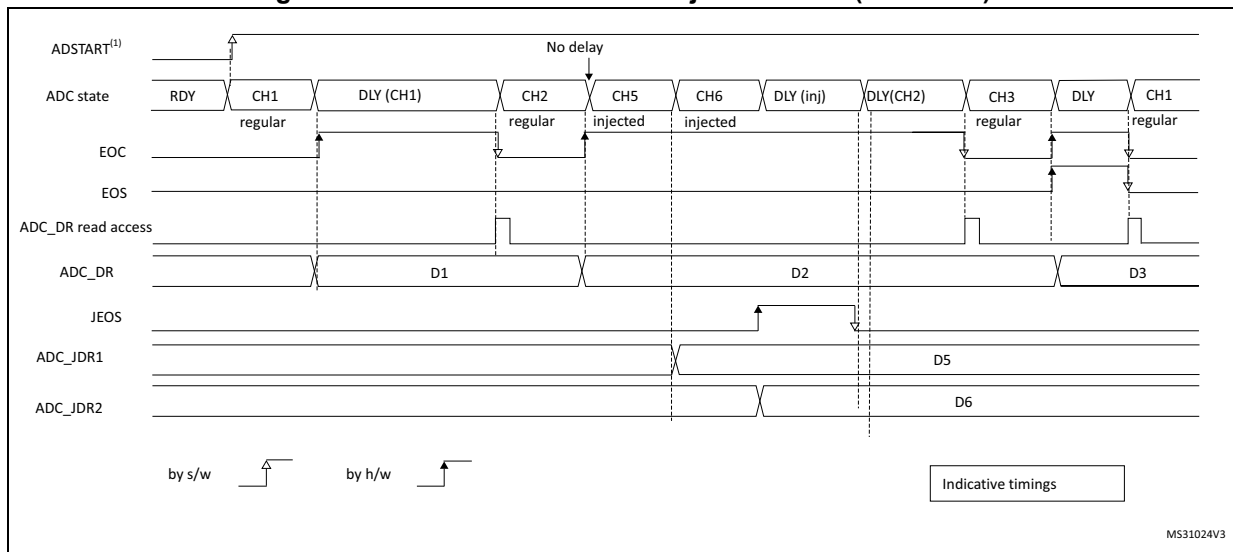
Figure 120. AUTODLY=1, regular continuous conversions interrupted by injected conversions



MS31023V3

1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, DISCEN=0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=0, CHANNELS = 5,6

Figure 121. AUTODLY=1 in auto-injected mode (JAUTO=1)



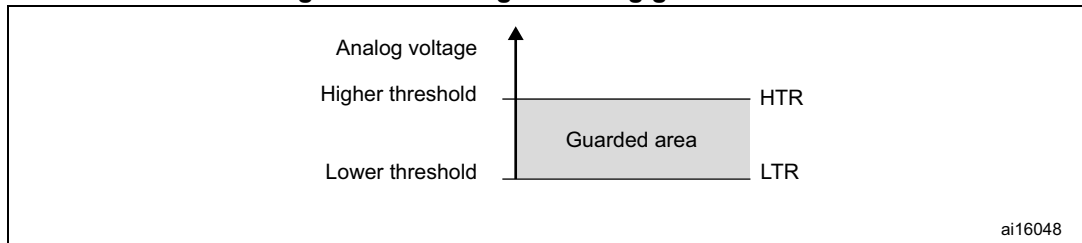
MS31024V3

1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, DISCEN=0, CHANNELS = 1, 2
3. Injected configuration: JAUTO=1, CHANNELS = 5,6

**21.4.29 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD\_HTx, AWD\_LTx, AWDx)**

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

**Figure 122. 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 ADCx\_IER register (x=1,2,3).

AWDx (x=1,2,3) flag is cleared by software by writing 1 to it.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

**Description of analog watchdog 1**

The AWD analog watchdog 1 is enabled by setting the AWD1EN bit in the ADCx\_CFGR register. This watchdog monitors whether either one selected channel or all enabled channels<sup>(1)</sup> remain within a configured voltage range (window).

Table 122 shows how the ADCx\_CFGR registers should be configured to enable the analog watchdog on one or more channels.

**Table 122. 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 ADCx\_TR1 register for the analog watchdog 1. When converting data with a resolution of less than 12 bits (according to bits RES[1:0]), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned).

Table 123 describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

**Table 123. Analog watchdog 1 comparison**

Resolution (bit RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned <sup>(1)</sup>	Thresholds	
00: 12-bit	DATA[11:0]	LT1[11:0] and HT1[11:0]	-
01: 10-bit	DATA[11:2],00	LT1[11:0] and HT1[11:0]	User must configure LT1[1:0] and HT1[1:0] to 00
10: 8-bit	DATA[11:4],0000	LT1[11:0] and HT1[11:0]	User must configure LT1[3:0] and HT1[3:0] to 0000
11: 6-bit	DATA[11:6],0000 00	LT1[11:0] and HT1[11:0]	User must configure LT1[5:0] and HT1[5:0] to 000000

1. The watchdog comparison is performed on the raw converted data before any alignment calculation and before applying any offsets (the data which is compared is not signed).

### Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDCHx[19:0] (x=2,3).

The corresponding watchdog is enabled when any bit of AWDCHx[19:0] (x=2,3) is set.

They are limited to a resolution of 8 bits and only the 8 MSBs of the thresholds can be programmed into HTx[7:0] and LTx[7:0]. Table 124 describes how the comparison is performed for all the possible resolutions.

**Table 124. Analog watchdog 2 and 3 comparison**

Resolution (bits RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned <sup>(1)</sup>	Thresholds	
00: 12-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	DATA[3:0] are not relevant for the comparison
01: 10-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	DATA[3:2] are not relevant for the comparison
10: 8-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	-
11: 6-bit	DATA[11:6],00	LTx[7:0] and HTx[7:0]	User must configure LTx[1:0] and HTx[1:0] to 00

1. The watchdog comparison is performed on the raw converted data before any alignment calculation and before applying any offsets (the data which is compared is not signed).

### ADCy\_AWDx\_OUT signal output generation

Each analog watchdog is associated to an internal hardware signal ADCy\_AWDx\_OUT (y=ADC number, x=watchdog number) which is directly connected to the ETR input (external trigger) of some on-chip timers. Refer to the on-chip timers section to understand how to select the ADCy\_AWDx\_OUT signal as ETR.

ADCy\_AWDx\_OUT is activated when the associated analog watchdog is enabled:

- ADCy\_AWDx\_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADCy\_AWDx\_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADCy\_AWDx\_OUT is also reset when disabling the ADC (when setting ADDIS=1). Note that stopping regular or injected conversions (setting ADSTP=1 or JADSTP=1) has no influence on the generation of ADCy\_AWDx\_OUT.

*Note: AWDx flag is set by hardware and reset by software: AWDx flag has no influence on the generation of ADCy\_AWDx\_OUT (ex: ADCy\_AWDx\_OUT can toggle while AWDx flag remains at 1 if the software did not clear the flag).*

**Figure 123. ADCy\_AWDx\_OUT signal generation (on all regular channels)**

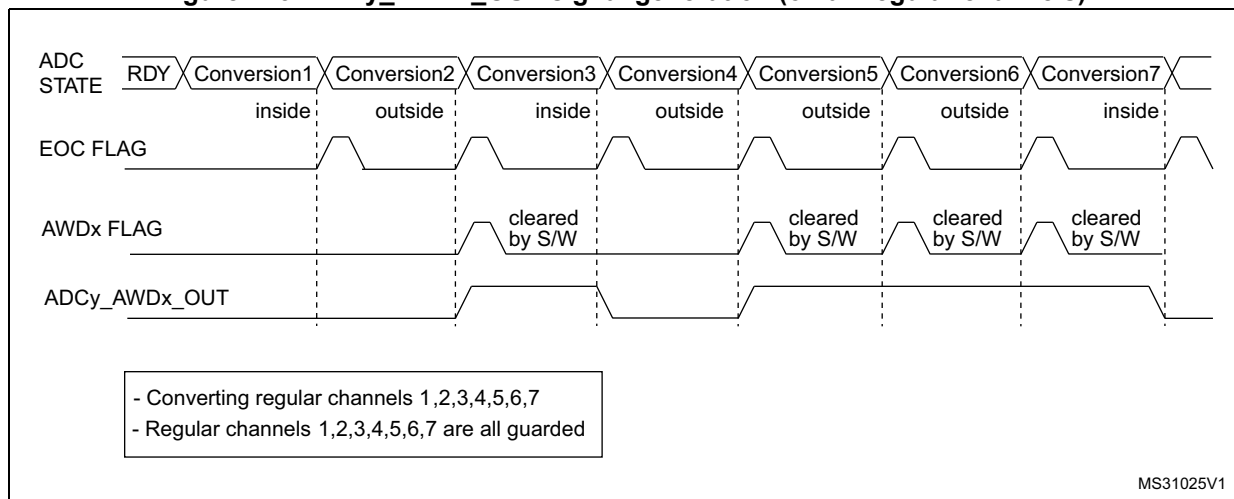


Figure 124. ADCy\_AWDx\_OUT signal generation (AWDx flag not cleared by software)

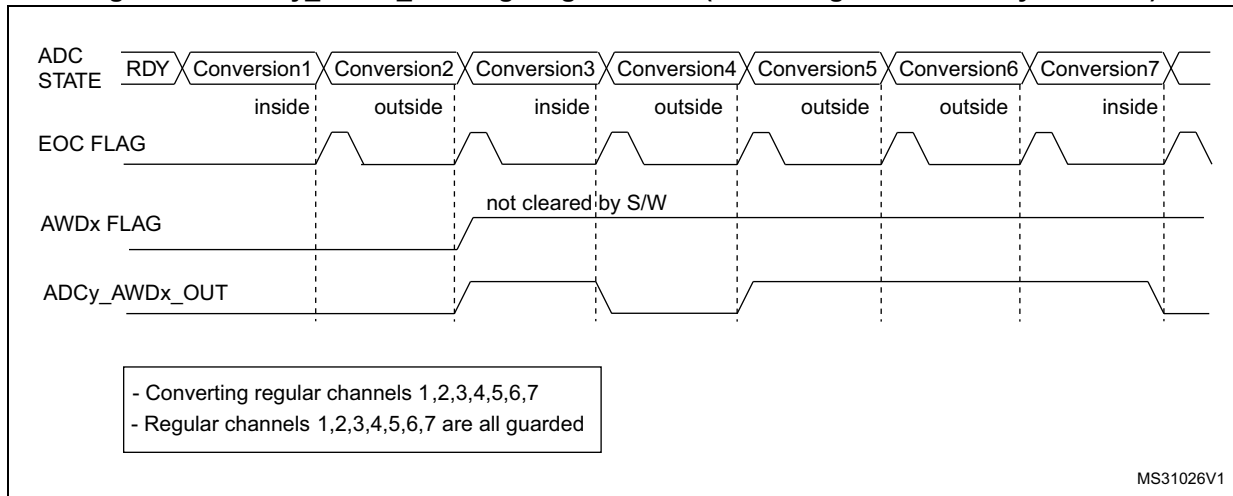


Figure 125. ADCy\_AWDx\_OUT signal generation (on a single regular channel)

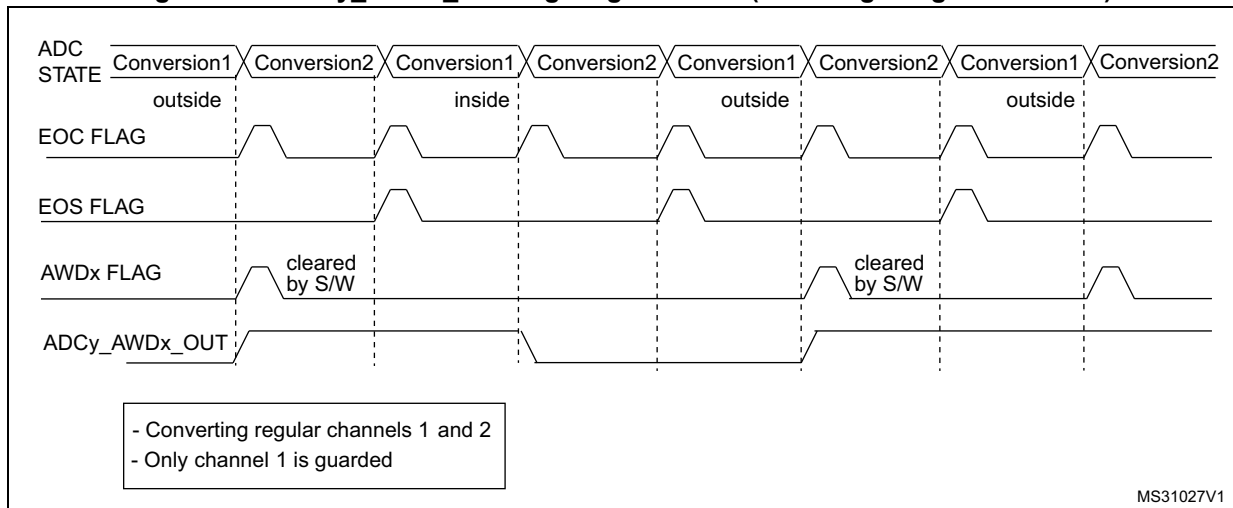
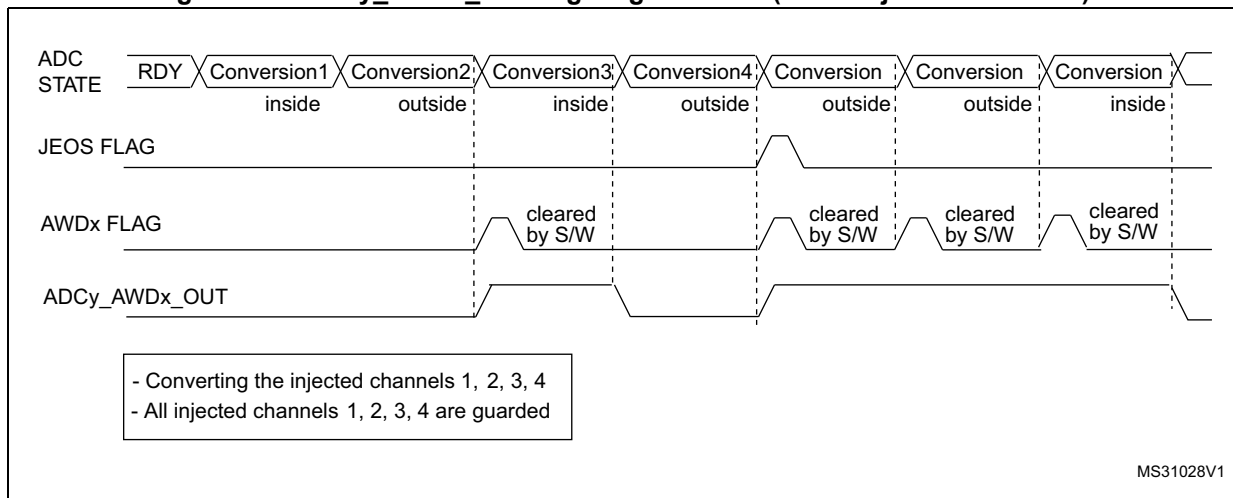


Figure 126. ADCy\_AWDx\_OUT signal generation (on all injected channels)



### 21.4.30 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:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

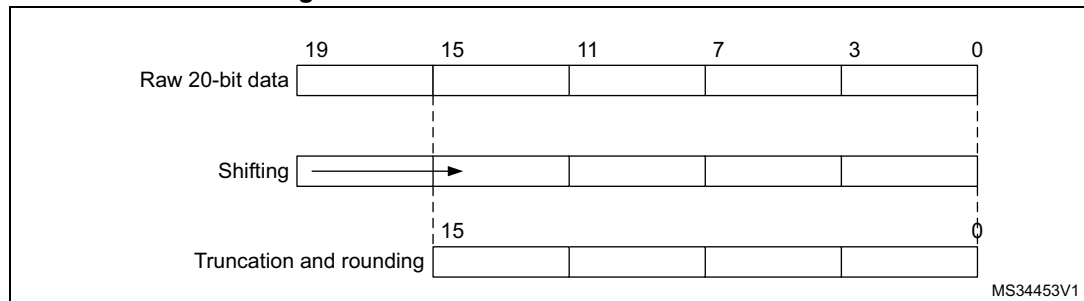
It allows to perform by hardware the following functions: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OVFS[2:0] bits in the ADCx\_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 ADCx\_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 ADCx\_DR data register.

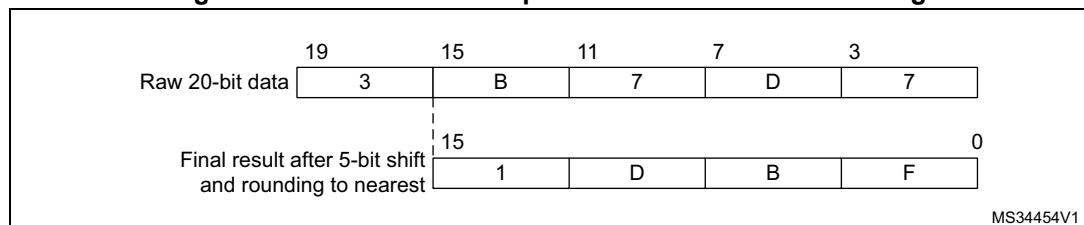
*Note: If the intermediary result after the shifting exceeds 16-bit, the result is truncated as is, without saturation.*

**Figure 127. 20-bit to 16-bit result truncation**



The [Figure 128](#) gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

**Figure 128. Numerical example with 5-bit shift and rounding**



[Table 125](#) gives the data format for the various N and M combinations, for a raw conversion data equal to 0xFFF.



**Table 125. 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	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
32x	0x1FFE0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200
64x	0x3FFC0	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
128x	0x7FF80	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800
256x	0xFFF00	0xFF00	0xFF80	0xFFC0	0xFFE0	0xFFF0	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 follow:

- 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 x 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 ADCx\_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 ADCx\_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 ADCx\_OFRRy register is ignored (considered as reset).*

### Analog watchdog

The analog watchdog functionality is maintained (AWDSGL and AWDEN bits), with the following difference:

- the RES[1:0] bits are ignored, comparison is always done on 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 ADCx\_DR[15:4]

*Note: Care must be taken when using high shifting values, this will reduce 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 ADCx\_DR[11:4] and HT[0:7] / LT[[0:7], and HT[11:8] / LT[11:8] must be kept reset.*

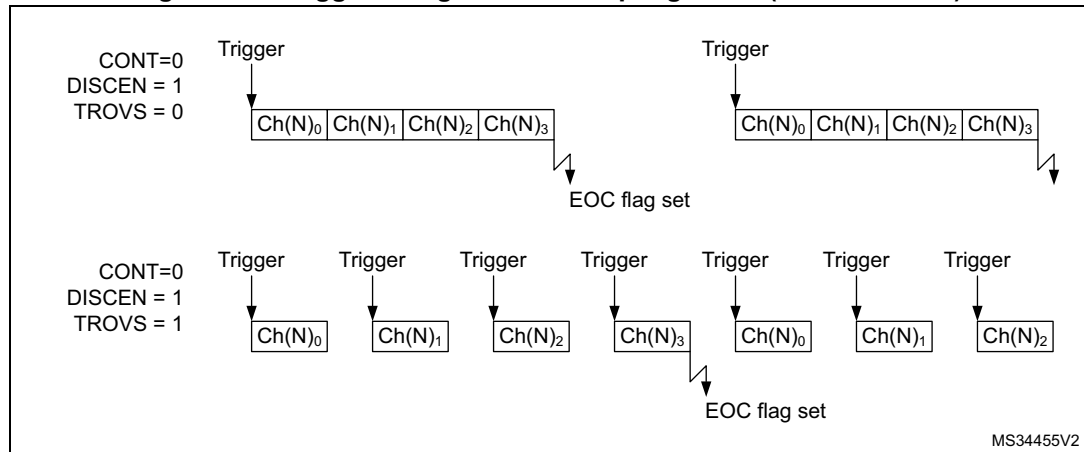
### Triggered mode

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 ADCx\_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

The [Figure 129](#) 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 129. 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).

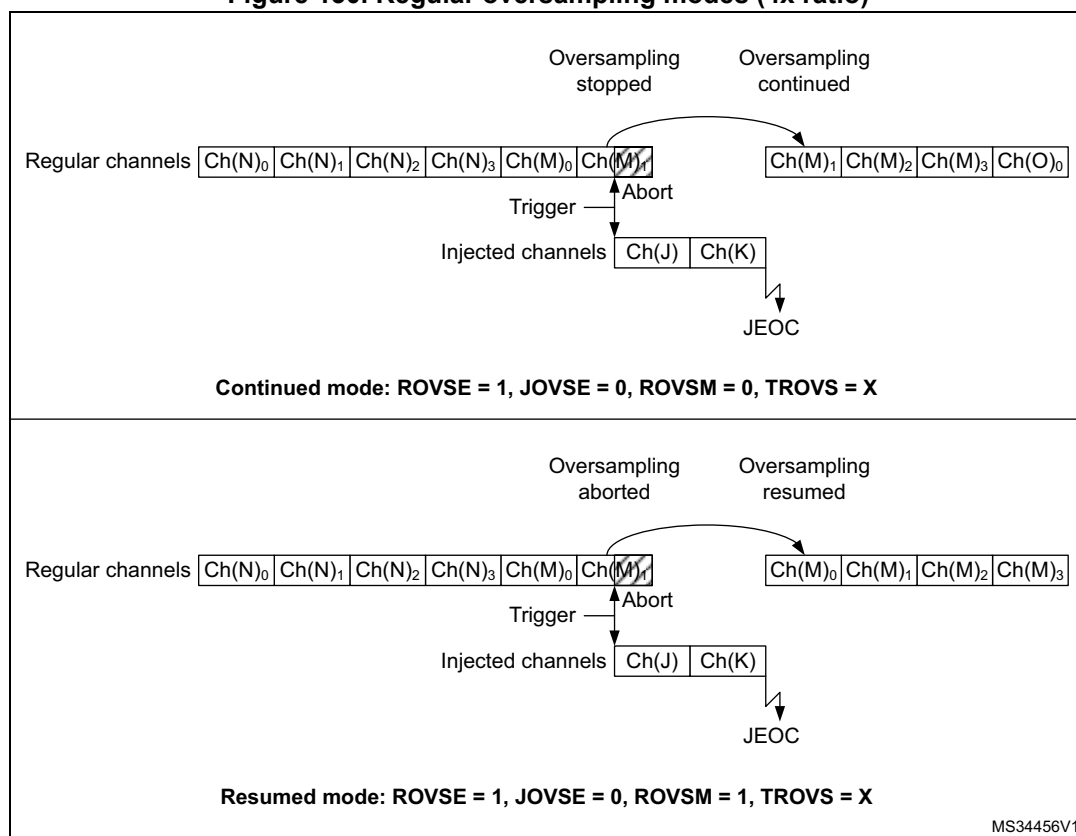
### Oversampling regular channels only

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 will be completed whatever the injection frequency (providing at least one regular conversion can be completed between triggers);
- in resumed mode, the accumulation restarts from 0 (previous conversions 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 completed and the regular sequencer will be blocked.

The [Figure 130](#) gives examples for a 4x oversampling ratio.

**Figure 130. Regular oversampling modes (4x ratio)**



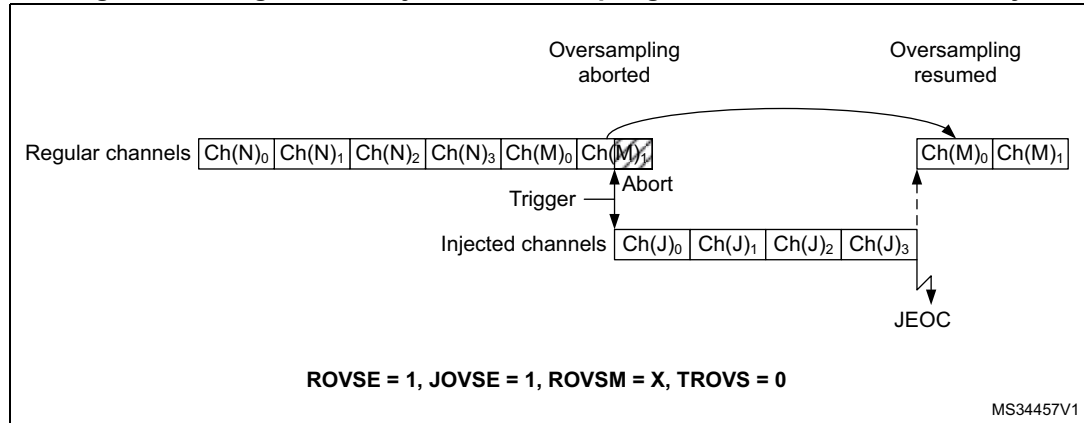
### Oversampling Injected channels only

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 131](#) below.

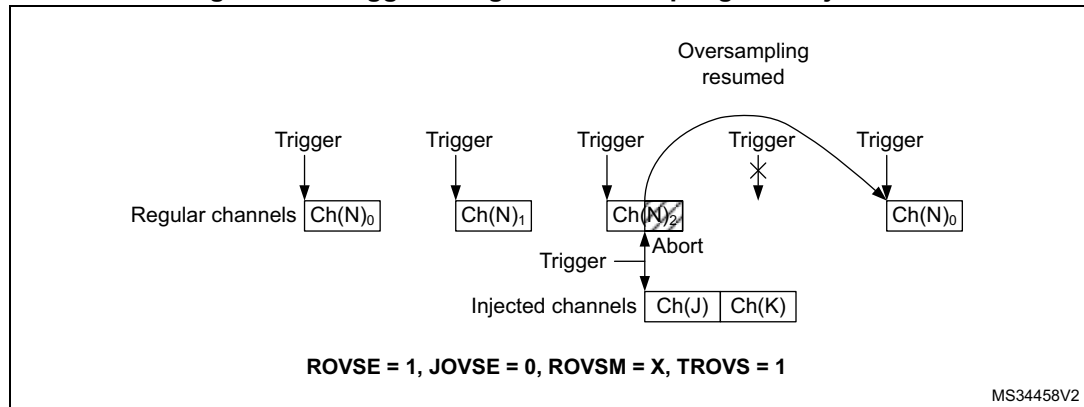
**Figure 131. Regular and injected oversampling modes used simultaneously**



### 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 132](#) below.

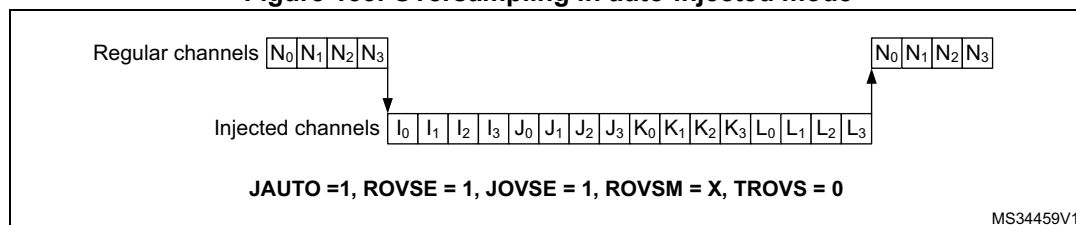
**Figure 132. Triggered regular oversampling with injection**



### 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 133](#) below shows how the conversions are sequenced.

**Figure 133. 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 TROVSE = 1.

### 21.4.31 Temperature sensor

The temperature sensor can be used to measure the junction temperature (Tj) 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 °C.

[Figure 134](#) shows the block diagram of connections between the temperature sensor and the ADC.

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation (up to 45 °C from one chip to another).

The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the temperature sensor measurement, calibration values are stored in system memory for each device by ST during production.

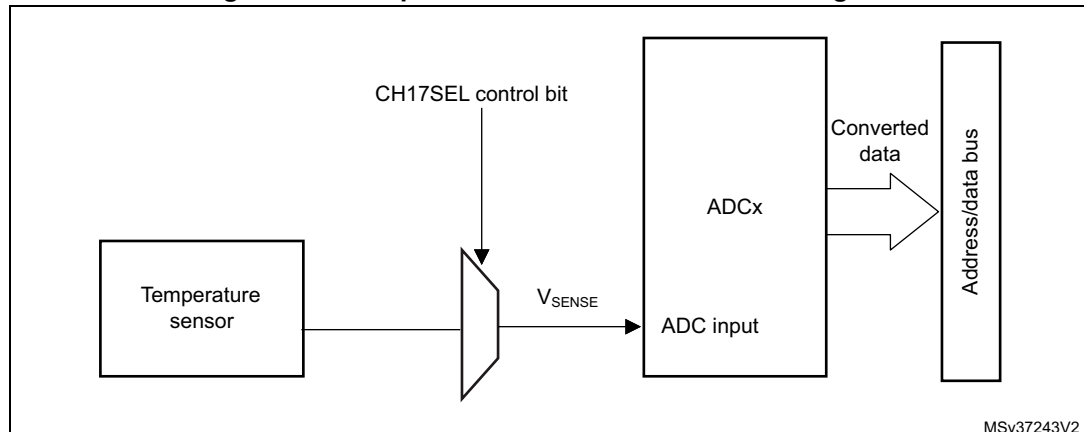
During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference (refer to the datasheet 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 134](#) shows the block diagram of the temperature sensor.

Figure 134. Temperature sensor channel block diagram



### Reading the temperature

To use the sensor:

1. Select the ADC input channels which is connected to  $V_{SENSE}$ .
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  $V_{SENSE}$  data in the ADC data register.
6. Calculate the actual temperature using the following formula:

$$\text{Temperature (in } ^\circ\text{C)} = \frac{110\text{ }^\circ\text{C} - 30\text{ }^\circ\text{C}}{\text{TS\_CAL2} - \text{TS\_CAL1}} \times (\text{TS\_DATA} - \text{TS\_CAL1}) + 30\text{ }^\circ\text{C}$$

Where:

- TS\_CAL2 is the temperature sensor calibration value acquired at 110°C
  - TS\_CAL1 is the temperature sensor calibration value acquired at 30°C
  - 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  $V_{SENSE}$  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.

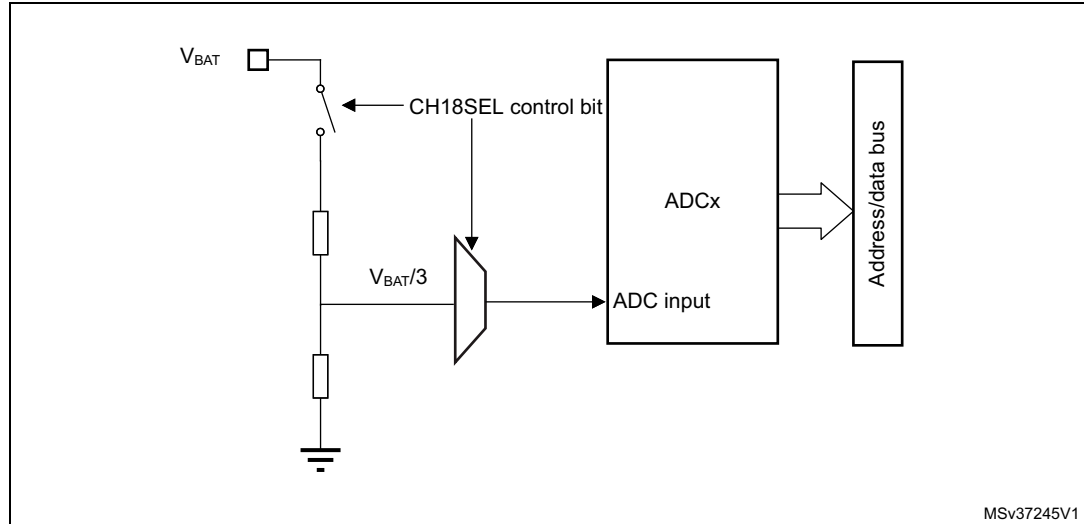
### 21.4.32 $V_{BAT}$ supply monitoring

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 135.  $V_{BAT}$  channel block diagram**



1. The CH18SEL bit must be set to enable the conversion of internal channel for  $V_{BAT}/3$ .

### 21.4.33 Monitoring the internal voltage reference

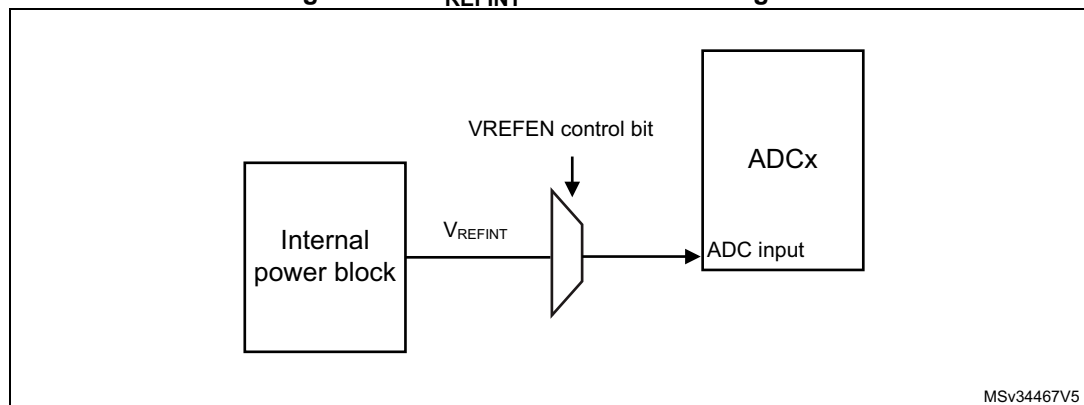
It is possible to monitor the internal voltage reference ( $V_{REFINT}$ ) to have a reference point for evaluating the ADC  $V_{REF+}$  voltage level.

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 136* shows the block diagram of the  $V_{REFINT}$  sensing feature.

**Figure 136.  $V_{REFINT}$  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 $V_{DDA}$ voltage using the internal reference voltage

The  $V_{DDA}$  power supply voltage applied to the microcontroller may be subject to variation or not precisely known. The embedded internal voltage reference ( $V_{REFINT}$ ) and its calibration data acquired by the ADC during the manufacturing process at  $V_{DDA} = 3.0$  V can be used to evaluate the actual  $V_{DDA}$  voltage level.

The following formula gives the actual  $V_{DDA}$  voltage supplying the device:

$$V_{DDA} = 3.0 \text{ V} \times V_{REFINT\_CAL} / V_{REFINT\_DATA}$$

where:

- $V_{REFINT\_CAL}$  is the  $V_{REFINT}$  calibration value
- $V_{REFINT\_DATA}$  is the actual  $V_{REFINT}$  output value converted by ADC

### Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between the analog power supply and the voltage applied on the converted channel. For most application use cases, it is necessary to convert this ratio into a voltage independent of  $V_{DDA}$ . For applications where  $V_{DDA}$  is known and ADC converted values are right-aligned you can use the following formula to get this absolute value:

$$V_{CHANNELx} = \frac{V_{DDA}}{FULL\_SCALE} \times ADCx\_DATA$$

For applications where  $V_{DDA}$  value is not known, you must use the internal voltage reference and  $V_{DDA}$  can be replaced by the expression provided in [Calculating the actual  \$V\_{DDA}\$  voltage using the internal reference voltage](#), resulting in the following formula:

$$V_{CHANNELx} = \frac{3.0 \text{ V} \times V_{REFINT\_CAL} \times ADCx\_DATA}{V_{REFINT\_DATA} \times FULL\_SCALE}$$

Where:

- $V_{REFINT\_CAL}$  is the  $V_{REFINT}$  calibration value
- $ADCx\_DATA$  is the value measured by the ADC on channel x (right-aligned)
- $V_{REFINT\_DATA}$  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 will be  $2^{12} - 1 = 4095$  or with 8-bit resolution,  $2^8 - 1 = 255$ .

**Note:** *If ADC measurements are done using an output format other than 12 bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.*



## 21.5 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 126. 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

## 21.6 ADC registers

Refer to [Section 1.2 on page 75](#) for a list of abbreviations used in register descriptions.

### 21.6.1 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	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	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 21.4.21: 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 ADCx\_TR3 register. It is cleared by software writing 1 to it.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 3 event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT2[7:0] and HT2[7:0] of ADCx\_TR2 register. It is cleared by software writing 1 to it.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 2 event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT1[11:0] and HT1[11:0] of ADCx\_TR1 register. It is cleared by software writing 1 to it.

0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 1 event occurred

Bit 6 **JEOS**: Injected channel end of sequence flag

This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it.

0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Injected conversions complete

**Bit 5 JEOC:** Injected channel end of conversion flag

This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADCx\_JDRy register. It is cleared by software writing 1 to it or by reading the corresponding ADCx\_JDRy register

0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Injected channel conversion complete

**Bit 4 OVR:** ADC overrun

This bit is set by hardware when an overrun occurs on a regular channel, meaning that a new conversion has completed while the EOC flag was already set. It is cleared by software writing 1 to it.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)

1: Overrun has occurred

**Bit 3 EOS:** End of regular sequence flag

This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it.

0: Regular Conversions sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Regular Conversions sequence complete

**Bit 2 EOC:** End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADCx\_DR register. It is cleared by software writing 1 to it or by reading the ADCx\_DR register

0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Regular channel conversion complete

**Bit 1 EOSMP:** End of sampling flag

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase.

0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

1: End of sampling phase reached

**Bit 0 ADRDY:** ADC ready

This bit is set by hardware after the ADC has been enabled (bit ADEN=1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

1: ADC is ready to start conversion

### 21.6.2 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	AWD3I E	AWD2I E	AWD1I E	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 JEOCIE:** End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.

0: JEOC interrupt disabled.

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

*Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no regular conversion is ongoing).*

**Bit 4 OVRIE:** Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

*Note: 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).*

### 21.6.3 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.	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 ADCx\_CALFACT only when ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing)*

**Bit 30 ADCALDIF:** Differential mode for calibration

This bit is set and cleared by software to configure the single-ended or differential inputs mode for the calibration.

0: Writing ADCAL will launch a calibration in single-ended inputs mode.

1: Writing ADCAL will launch a calibration in differential inputs mode.

*Note: 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 21.4.6: 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, a conversion will start immediately (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

- in single conversion mode when software trigger is selected (JEXTSEL=0x0): at the assertion of the End of Injected Conversion Sequence (JEOS) flag.
- in all cases: after the execution of the JADSTP command, at the same time that JADSTP is cleared by hardware.

0: No ADC injected conversion is ongoing.

1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

*Note: 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 EXTEN, a conversion will start immediately (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

- in single conversion mode when software trigger is selected (EXTSEL=0x0): at the assertion of the End of Regular Conversion Sequence (EOS) flag.
  - in all cases: after the execution of the ADSTP command, at the same time that ADSTP is cleared by hardware.
- 0: No ADC regular conversion is ongoing.  
1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

*Note: 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 will be effectively ready to operate once the flag ADRDY has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

*Note: The software is allowed to set ADEN only when all bits of ADCx\_CR registers are 0 (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0) except for bit ADVREGEN which must be 1 (and the software must have wait for the startup time of the voltage regulator)*



### 21.6.4 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	JAWD1 EN	AWD1 EN	AWD1S GL	JQM	JDISC EN	DISCNUM[2:0]			DISC EN	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AUT DLY	CONT	OVR MOD	EXTEN[1:0]		EXTSE L3	EXTSE L2	EXTSE L1	EXTSE L0	ALIGN	RES[1:0]		DFSD MCFG	DMA CFG	DMA EN
	rw	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 (available on ADC1 only)

00001: ADC analog input channel 1 monitored by AWD1

.....

10010: ADC analog input channel 18 monitored by AWD1

others: reserved, must not be used

*Note: The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers.*

*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 21.4.21: 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: ADCx\_DR register is preserved with the old data when an overrun is detected.

1: ADCx\_DR register is overwritten with the last conversion result when an overrun is detected.

*Note: 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 \(ADCx\\_DR, OFFSETy, OFFSETy\\_CH, ALIGN\)](#)

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 **DFSDMCFG**: DFSDM mode configuration

This bit is set and cleared by software to enable the DFSDM mode. It is effective only when DMAEN=0.

- 0: DFSDM mode disabled
- 1: DFSDM mode enabled

*Note: To make sure no conversion is ongoing, the software is allowed to write this bit only when ADSTART= 0 and JADSTART= 0.*

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 GP-DMA to manage automatically the converted data. For more details, refer to [Section : Managing conversions using the DMA](#).

- 0: DMA disabled
- 1: DMA enabled

*Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).*

### 21.6.5 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 injected 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 injected 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)*

### 21.6.6 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	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
SMP5_0	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:0 **SMPx[2:0]**: Channel x sampling time selection

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

- 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).*

### 21.6.7 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

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).*

### 21.6.8 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]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LT1[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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 21.4.29: 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 15:12 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 21.4.29: 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).*

### 21.6.9 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:16 **HT2[7:0]**: Analog watchdog 2 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 2.

Refer to [Section 21.4.29: 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 21.4.29: 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).*

### 21.6.10 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT3[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT3[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:16 **HT3[7:0]**: Analog watchdog 3 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 3.

Refer to [Section 21.4.29: 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).*

### 21.6.11 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		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ2[3:0]			Res.	SQ1[4:0]					Res.	Res.	L[3:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]**: 4th conversion in regular sequence

These bits are written by software with the channel number (0..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..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..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..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).*

### 21.6.12 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..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..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..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..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..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).*

### 21.6.13 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..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..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..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..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..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).*

### 21.6.14 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
					rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

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

Bits 10:6 **SQ16[4:0]**: 16th conversion in regular sequence

These bits are written by software with the channel number (0..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..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).*

### 21.6.15 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 21.4.26: Data management](#).

### 21.6.16 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..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..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..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..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 at any time, once the ADC is enabled (ADEN=1).*

*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 21.4.21: 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 at any time, once the ADC is enabled (ADEN=1).*

Bits 1:0 **JL[1:0]**: Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

*Note: The software is allowed to write these bits at any time, once the ADC is enabled (ADEN=1).*

### 21.6.17 ADC offset y register (ADC\_OFRRy)

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
OFFSETy_EN	OFFSETy_CH[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r/w	r/w	r/w	r/w	r/w	r/w										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	OFFSETy[11:0]											
				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w



Bit 31 **OFFSETy\_EN**: Offset y Enable

This bit is written by software to enable or disable the offset programmed into bits OFFSETy[11:0].

*Note: 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 **OFFSETy\_CH[4:0]**: Channel selection for the Data offset y

These bits are written by software to define the channel to which the offset programmed into bits OFFSETy[11:0] will apply.

*Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).*

Bits 25:12 Reserved, must be kept at reset value.

Bits 11:0 **OFFSETy[11:0]**: Data offset y for the channel programmed into bits OFFSETy\_CH[4:0]

These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset y must be programmed in the bits OFFSETy\_CH[4:0]. The conversion result can be read from in the ADCx\_DR (regular conversion) or from in the ADCx\_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.*

### 21.6.18 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 21.4.26: Data management](#).

### 21.6.19 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.	AWD2CH[18:16]		
													r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD2CH[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:19 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).*

### 21.6.20 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.	AWD3CH[18:16]		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD3CH[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:19 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 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).*

### 21.6.21 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.	DIFSEL[18:16]		
													r	r	r
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	r

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

Bits 18:16 **DIFSEL[18:16]**: Differential mode for channels 18 to 16.

These bits are read only. These channels are forced to single-ended input mode (either connected to a single-ended I/O port or to an internal channel).

Bits 15:1 **DIFSEL[15:1]**: Differential mode for channels 15 to 1

These bits are set and cleared by software. They allow to select if a channel is configured as single ended or differential mode.

DIFSEL[i] = 0: ADC analog input channel i is configured in single ended mode

DIFSEL[i] = 1: ADC analog input channel i is configured in differential mode

*Note: 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).*

Bit 0 **DIFSEL[0]**: Differential mode for channel 0

This bit is read only. This channel is forced to single-ended input mode (connected to an internal channel).

### 21.6.22 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_D[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_S[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

Bits 22:16 **CALFACT\_D[6:0]**: Calibration Factors in differential mode

These bits are written by hardware or by software.

Once a differential inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it will then be applied once a new differential calibration is launched.

*Note: 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 will then be 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).*

## 21.7 ADC common registers

These registers define the control and status registers common to master and slave ADCs:

### 21.7.1 ADC common status register (ADC\_CSR)

Address offset: 0x00 (this offset address is relative to the master ADC base address + 0x300)

Reset value: 0x0000 0000

This register provides an image of the status bits of the different ADCs. Nevertheless it is read-only and does not allow to clear the different status bits. Instead each status bit must be cleared by writing 0 to it in the corresponding ADCx\_ISR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADDRDY_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 ADCx\_ISR register.

Bit 9 **AWD3\_MST**: Analog watchdog 3 flag of the master ADC

This bit is a copy of the AWD3 bit in the corresponding ADCx\_ISR register.

- Bit 8 **AWD2\_MST**: Analog watchdog 2 flag of the master ADC  
This bit is a copy of the AWD2 bit in the corresponding ADCx\_ISR register.
- Bit 7 **AWD1\_MST**: Analog watchdog 1 flag of the master ADC  
This bit is a copy of the AWD1 bit in the corresponding ADCx\_ISR register.
- Bit 6 **JEOS\_MST**: End of injected sequence flag of the master ADC  
This bit is a copy of the JEOS bit in the corresponding ADCx\_ISR register.
- Bit 5 **JEOC\_MST**: End of injected conversion flag of the master ADC  
This bit is a copy of the JEOC bit in the corresponding ADCx\_ISR register.
- Bit 4 **OVR\_MST**: Overrun flag of the master ADC  
This bit is a copy of the OVR bit in the corresponding ADCx\_ISR register.
- Bit 3 **EOS\_MST**: End of regular sequence flag of the master ADC  
This bit is a copy of the EOS bit in the corresponding ADCx\_ISR register.
- Bit 2 **EOC\_MST**: End of regular conversion of the master ADC  
This bit is a copy of the EOC bit in the corresponding ADCx\_ISR register.
- Bit 1 **EOSMP\_MST**: End of Sampling phase flag of the master ADC  
This bit is a copy of the EOSMP bit in the corresponding ADCx\_ISR register.
- Bit 0 **ADRDY\_MST**: Master ADC ready  
This bit is a copy of the ADRDY bit in the corresponding ADCx\_ISR register.

### 21.7.2 ADC common control register (ADC\_CCR)

Address offset: 0x08 (this offset address is relative to the master ADC base address + 0x300)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH18SEL	CH17SEL	VREFEN	PRESC[3:0]				CKMODE[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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

- Bit 24 **CH18SEL**: CH18 selection  
This bit is set and cleared by software to control channel 18  
0: V<sub>BAT</sub> channel disabled.  
1: V<sub>BAT</sub> 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<sub>REFINT</sub> enable

This bit is set and cleared by software to enable/disable the V<sub>REFINT</sub> channel.

0: V<sub>REFINT</sub> channel disabled

1: V<sub>REFINT</sub> 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] = 0b00.*

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 to 1 (HPRE[3:0] = 0xxx in RCC\_CFGR register) and if the system clock has a 50% duty cycle.

10: HCLK/2 (Synchronous clock mode)

11: HCLK/4 (Synchronous clock mode)

In all synchronous clock modes, there is no jitter in the delay from a timer trigger to the start of a conversion.

*Note: 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.

### 21.7.3 ADC register map

Table 127. ADC 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	ADC_ISR	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
	Reset value																						0	0	0	0	0	0	0	0	0	0	0	0

Table 127. 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
0x04	ADC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOGIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x08	ADC_CR	ADCAL	ADCALDIF	DEEPPWD	ADVREGEN																						JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN	
	Reset value	0	0	1	0																						0	0	0	0	0	0	0
0x0C	ADC_CFGR	JQDIS	AWD1CH[4:0]				JAUTO	JAWDTEN	AWD1TEN	AWD1SGL	JQM	JDISCEN	DISCNUM [2:0]		DISCEN			AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL3	EXTSEL2	EXTSEL1	EXTSEL0	ALIGN	RES [1:0]		DFSDMCFG	DMACFG	DMAEN	
	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
0x0C	ADC_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ROVSM	TROVS	OVSS[3:0]			OVSR [2:0]		JOVSE	ROVSE			
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x14	ADC_SMPR1	Res.	Res.	SMP9 [2:0]		SMP8 [2:0]		SMP7 [2:0]		SMP6 [2:0]		SMP5 [2:0]		SMP4 [2:0]		SMP3 [2:0]		SMP2 [2:0]		SMP1 [2:0]		SMP0 [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									
0x18	ADC_SMPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	Reserved	Res.																															
0x20	ADC_TR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	ADC_TR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	ADC_TR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																
0x2C	Reserved	Res.																															
0x30	ADC_SQR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	ADC_SQR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	ADC_SQR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	ADC_SQR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	ADC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																
0x44-0x48	Reserved	Res.																															
0x4C	ADC_JSQR	Res.	JSQ4[4:0]				Res.	JSQ3[4:0]				Res.	JSQ2[4:0]				Res.	JSQ1[4:0]				JEXTEN[1:0]	JEXTSEL [3:0]		JL[1:0]								
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x50-0x5C	Reserved	Res.																															



Table 127. 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						
0x60	ADC_OFR1	OFFSET1_EN	OFFSET1_CH[4:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET1[11:0]																
	Reset value	0	0	0	0	0	0																0	0	0	0	0	0	0	0	0	0	0	0					
0x64	ADC_OFR2	OFFSET2_EN	OFFSET2_CH[4:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET2[11:0]															
	Reset value	0	0	0	0	0	0																0	0	0	0	0	0	0	0	0	0	0	0					
0x68	ADC_OFR3	OFFSET3_EN	OFFSET3_CH[4:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET3[11:0]															
	Reset value	0	0	0	0	0	0																0	0	0	0	0	0	0	0	0	0	0	0					
0x6C	ADC_OFR4	OFFSET4_EN	OFFSET4_CH[4:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET4[11:0]															
	Reset value	0	0	0	0	0	0																0	0	0	0	0	0	0	0	0	0	0	0					
0x70-0x7C	Reserved	Res.																																					
0x80	ADC_JDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA1[15:0]																					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0x84	ADC_JDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA2[15:0]																					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0x88	ADC_JDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA3[15:0]																					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0x8C	ADC_JDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA4[15:0]																					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0x8C-0x9C	Reserved	Res.																																					
0xA0	ADC_AWD2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2CH[18:0]																					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0xA4	ADC_AWD3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3CH[18:0]																					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0xA8-0xAC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						
0xB0	ADC_DIFSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[18:0]																					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0xB4	ADC_CALFACT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_D[6:0]						Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_S[6:0]									
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						



**Table 128. 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.	0	0	0	0	0	0	0	0	0	0	0
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x04	Reserved	Res.																															
0x08	ADC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH18SEL	CH17SEL	VREFEN	PRESC[3:0]				CKMODE[1:0]		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															

Refer to [Section 2.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

## 22 Digital-to-analog converter (DAC)

### 22.1 Introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data could be left- or right-aligned. The DAC has two output channels, each with its own converter. In dual DAC channel mode, conversions could be done independently or simultaneously when both channels are grouped together for synchronous update operations. An input reference pin,  $V_{REF+}$  (shared with others analog peripherals) is available for better resolution. An internal reference can also be set on the same input. Refer to *voltage reference buffer (VREFBUF)* section.

The DAC\_OUTx pin can be used as general purpose input/output (GPIO) when the DAC output is disconnected from output pad and connected to on chip peripheral. The DAC output buffer can be optionally enabled to allow a high drive output current. An individual calibration can be applied on each DAC output channel. The DAC output channels support a low power mode, the Sample and Hold mode.

### 22.2 DAC main features

The DAC main features are the following (see [Figure 137: Dual-channel DAC block diagram](#))

- One DAC interface, maximum two output channels
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel including DMA underrun error detection
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- Each DAC output can be disconnected from the DAC\_OUTx output pin
- DAC output connection to on chip peripherals
- Sample and Hold mode for low power operation in Stop mode
- Input voltage reference,  $V_{REF+}$

[Figure 137](#) shows the block diagram of a DAC channel and [Table 130](#) gives the pin description.

## 22.3 DAC implementation

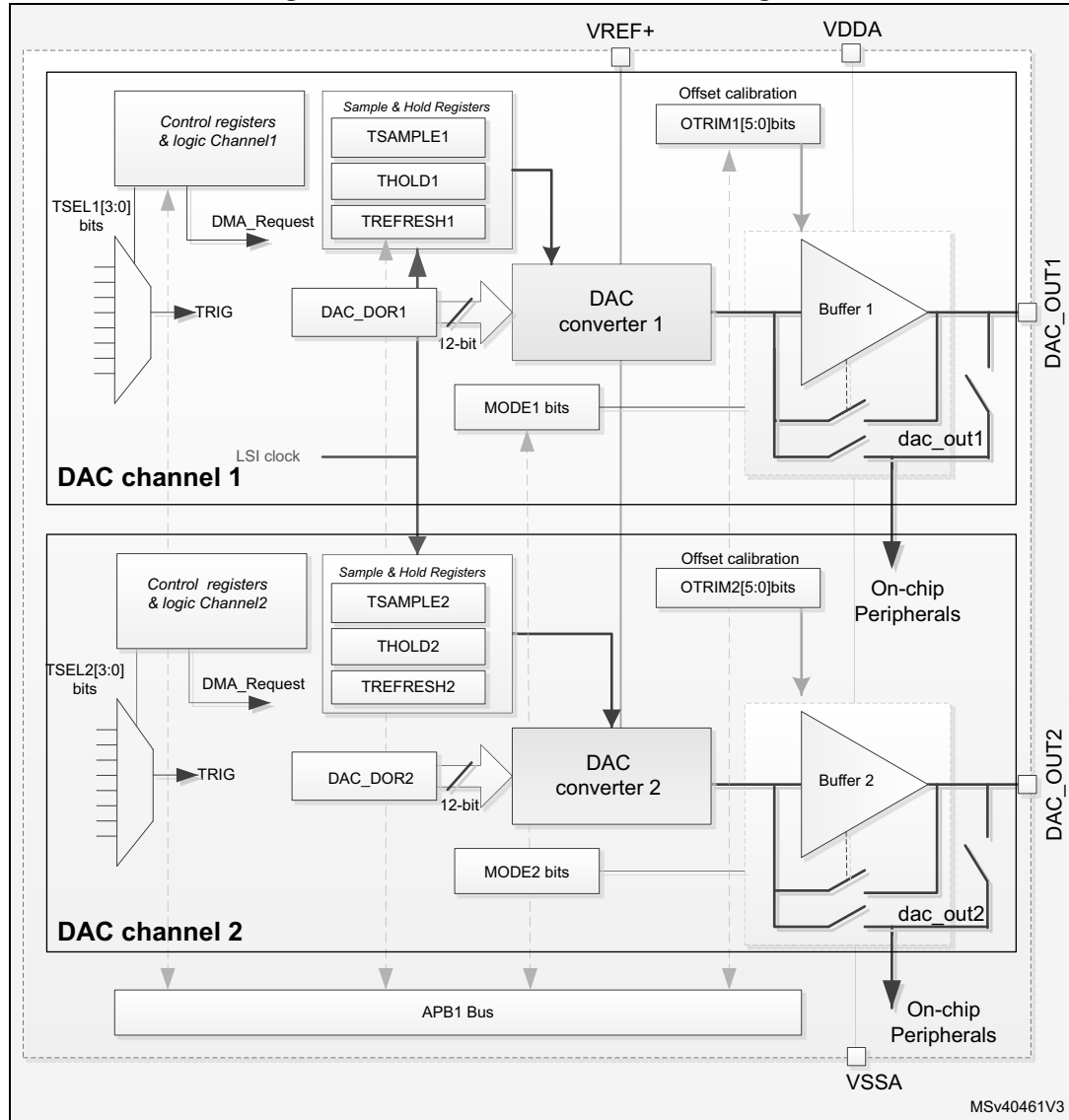
Table 129. DAC implementation

DAC features	DAC1
Dual channel	X
I/O connection	DAC1_OUT1 on PA4, DAC1_OUT2 on PA5

## 22.4 DAC functional description

### 22.4.1 DAC block diagram

Figure 137. Dual-channel DAC block diagram



1. MODEx bits in the DAC\_MCR control the output mode and allow switching between the Normal mode in buffer/unbuffered configuration and the Sample and Hold mode.
2. Refer to [Section 22.3: DAC implementation](#) for channel 2 availability.

The DAC includes:

- Up to two output channels
- The DAC\_OUTx can be disconnected from the output pin and used as an ordinary GPIO
- The DAC\_OUTx can use an internal pin connection to on-chip peripherals such as comparator, operational amplifier and ADC (if available).
- DAC output channel buffered or non buffered
- Sample and Hold block and registers operational in Stop mode, using LSI clock source for static conversion

The DAC includes up to two separate output channels. Each output channel can be connected to on-chip peripherals such as comparator, operational amplifier and ADC (if available). In this case, the DAC output channel can be disconnected from the DAC\_OUTx output pin and the corresponding GPIO can be used for another purpose.

The DAC output can be buffered or not. The Sample and Hold block and its associated registers can run in Stop mode using the LSI clock source.

**Table 130. DAC input/output pins**

Pin name	Signal type	Remarks
V <sub>REF+</sub>	Input, analog reference positive	The higher/positive reference voltage for the DAC, $V_{REF+} \leq V_{DDAmax}$ (refer to datasheet)
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply
DAC_OUTx	Analog output signal	DAC channelx analog output

### 22.4.2 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC\_CR register. The DAC channel is then enabled after a  $t_{WAKEUP}$  startup time.

*Note:* The ENx bit enables the analog DAC Channelx only. The DAC Channelx digital interface is enabled even if the ENx bit is reset.

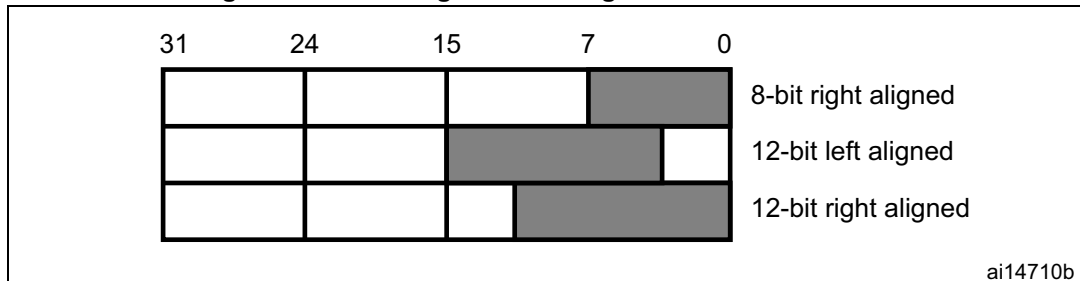
### 22.4.3 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channel
  - There are three possibilities:
    - 8-bit right alignment: the software has to load data into the DAC\_DHR8Rx[7:0] bits (stored into the DHRx[11:4] bits)
    - 12-bit left alignment: the software has to load data into the DAC\_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)
    - 12-bit right alignment: the software has to load data into the DAC\_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

Depending on the loaded DAC\_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DHRx (data holding registerx, which are internal non-memory-mapped registers). The DHRx register is then loaded into the DORx register either automatically, by software trigger or by an external event trigger.

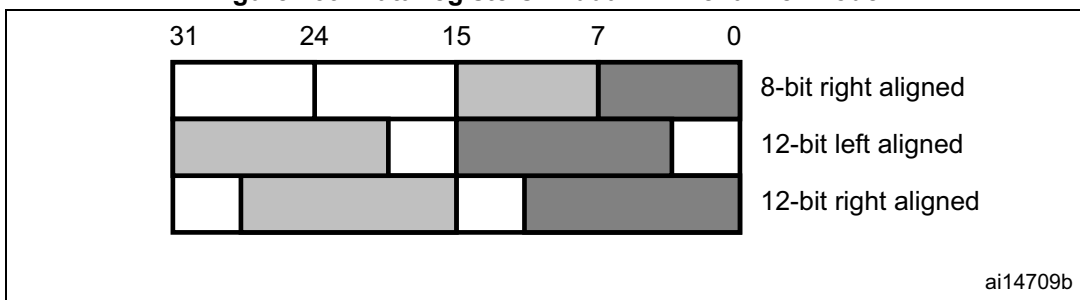
**Figure 138. Data registers in single DAC channel mode**



- Dual DAC channels (when available)  
 There are three possibilities:
  - 8-bit right alignment: data for DAC channel1 to be loaded into the DAC\_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC\_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)
  - 12-bit left alignment: data for DAC channel1 to be loaded into the DAC\_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC\_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)
  - 12-bit right alignment: data for DAC channel1 to be loaded into the DAC\_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC\_DHR12RD [27:16] bits (stored into the DHR2[11:0] bits)

Depending on the loaded DAC\_DHRyyyD register, the data written by the user is shifted and stored into DHR1 and DHR2 (data holding registers, which are internal non-memory-mapped registers). The DHR1 and DHR2 registers are then loaded into the DAC\_DOR1 and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

**Figure 139. Data registers in dual DAC channel mode**



### 22.4.4 DAC conversion

The DAC\_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC\_DHRx register (write operation to DAC\_DHR8Rx, DAC\_DHR12Lx, DAC\_DHR12Rx, DAC\_DHR8RD, DAC\_DHR12RD or DAC\_DHR12LD).

Data stored in the DAC\_DHRx register are automatically transferred to the DAC\_DORx register after one APB1 clock cycle, if no hardware trigger is selected (TENx bit in DAC\_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC\_CR register is set) and a trigger occurs, the transfer is performed three APB1 clock cycles after the trigger signal.

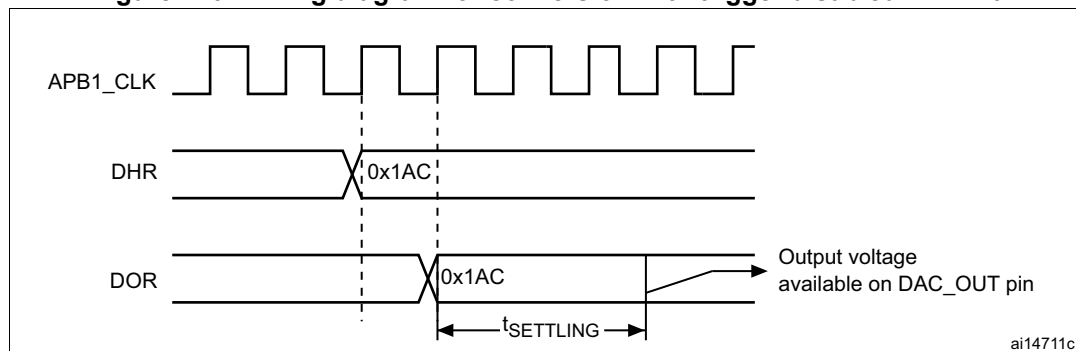
When DAC\_DORx is loaded with the DAC\_DHRx contents, the analog output voltage becomes available after a time  $t_{SETTLING}$  that depends on the power supply voltage and the analog output load.

HFSEL bit of DAC\_CR must be set when APB1 clock speed is faster than 80 MHz. It adds an extra delay of three APB1 clock cycles to the transfer from DAC\_DHRx register to DAC\_DORx register ( $t_{SETTLING}$ ).

The DAC\_DORx update rate is limited to 1/3 of APB1 clock frequency. When HFSEL bit is set, this rate is limited to 1/8 of the APB1 clock frequency.

When HFSEL is set, it is not allowed to write the DHRx register during a period of eight clock cycles after the ENx bit is set. During this period, making software/hardware triggering is not allowed either.

Figure 140. Timing diagram for conversion with trigger disabled TEN = 0



### 22.4.5 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and  $V_{REF+}$ .

The analog output voltages on each DAC channel pin are determined by the following equation:

$$DAC_{output} = V_{REF} \times \frac{DOR}{4096}$$

## 22.4.6 DAC trigger selection

If the TENx control bit is set, conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[2:0] control bits determine which out of 8 possible events will trigger conversion as shown in bits TSEL1[2:0] and TSEL2[2:0] in [Table 131: DAC trigger selection](#).

Each time a DAC interface detects a rising edge on the selected trigger source (refer to the table below), the last data stored into the DAC\_DHRx register are transferred into the DAC\_DORx register. The DAC\_DORx register is updated three APB1 cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC\_DORx register has been loaded with the DAC\_DHRx register contents.

*Note:* TSELx[2:0] bit cannot be changed when the ENx bit is set.

*When software trigger is selected, the transfer from the DAC\_DHRx register to the DAC\_DORx register takes only one APB1 clock cycle.*

**Table 131. DAC trigger selection**

Source	Type	TSELx[3:0]
SWTRIG	Software control bit	0000
TIM1_TRGO	Internal signal from on-chip timers	0001
TIM2_TRGO	Internal signal from on-chip timers	0010
TIM4_TRGO	Internal signal from on-chip timers	0011
TIM5_TRGO	Internal signal from on-chip timers	0100
TIM6_TRGO	Internal signal from on-chip timers	0101
TIM7_TRGO	Internal signal from on-chip timers	0110
TIM8_TRGO	Internal signal from on-chip timers	0111
TIM15_TRGO	Internal signal from on-chip timers	1000
Reserved	-	1001
Reserved	-	1010
LPTIM1_OUT	Internal signal from on-chip timers	1011
LPTIM2_OUT	Internal signal from on-chip timers	1100
EXTI9	External pin	1101
Reserved	-	1110
Reserved	-	1111



### 22.4.7 DMA requests

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

When an external trigger (but not a software trigger) occurs while the DMAENx bit is set, the value of the DAC\_DHRx register is transferred into the DAC\_DORx register when the transfer is complete, and a DMA request is generated.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, only the corresponding DMAENx bit should be set. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

As DAC\_DHRx to DAC\_DORx data transfer occurred before the DMA request, the very first data has to be written to the DAC\_DHRx before the first trigger event occurs.

#### DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC\_SR register is set, reporting the error condition. The DAC channelx continues to convert old data.

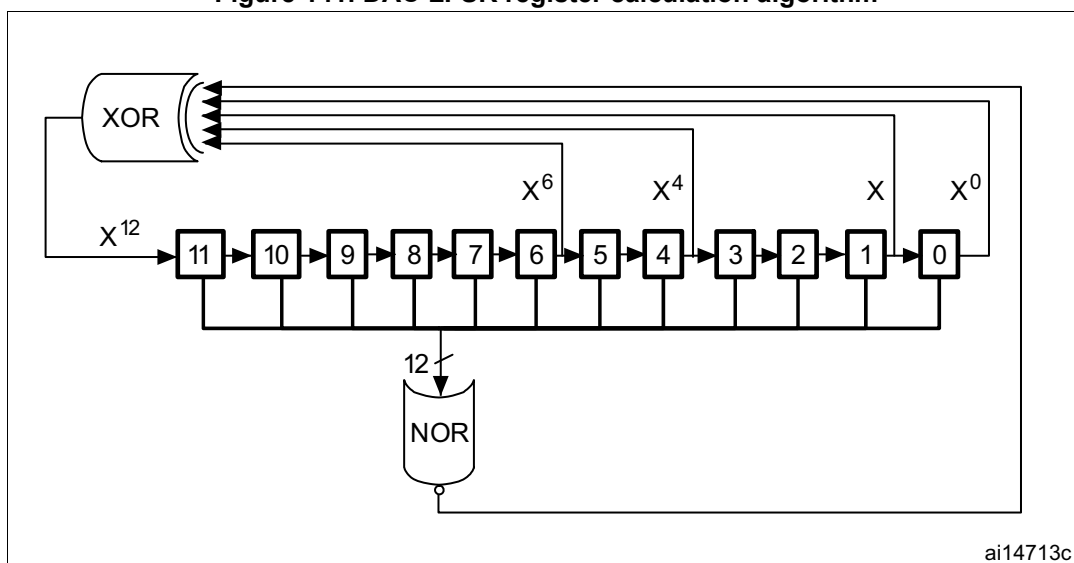
The software should clear the DMAUDRx flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software should modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion could be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channelx, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC\_CR register is enabled.

### 22.4.8 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to 01". The preloaded value in LFSR is 0xAAA. This register is updated three APB1 clock cycles after each trigger event, following a specific calculation algorithm.

Figure 141. DAC LFSR register calculation algorithm

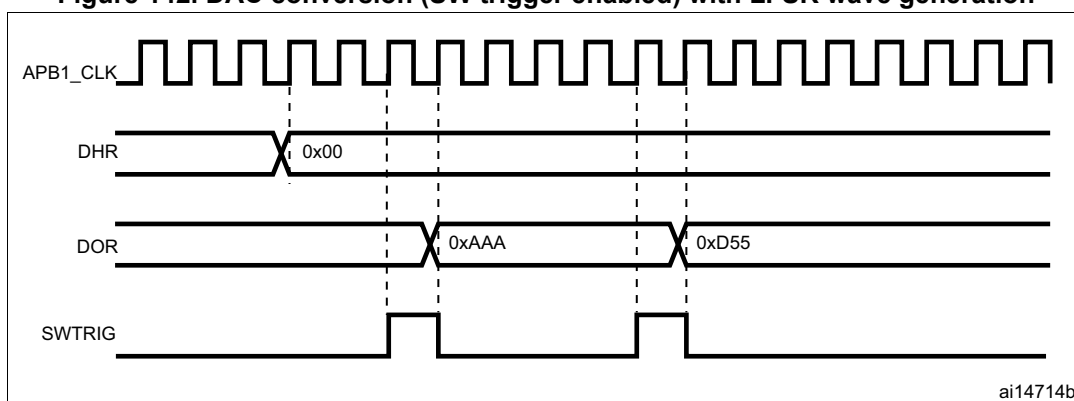


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC\_CR register, is added up to the DAC\_DHRx contents without overflow and this value is then transferred into the DAC\_DORx register.

If LFSR is 0x0000, a '1' is injected into it (antiloop-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Figure 142. DAC conversion (SW trigger enabled) with LFSR wave generation



*Note:* The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC\_CR register.

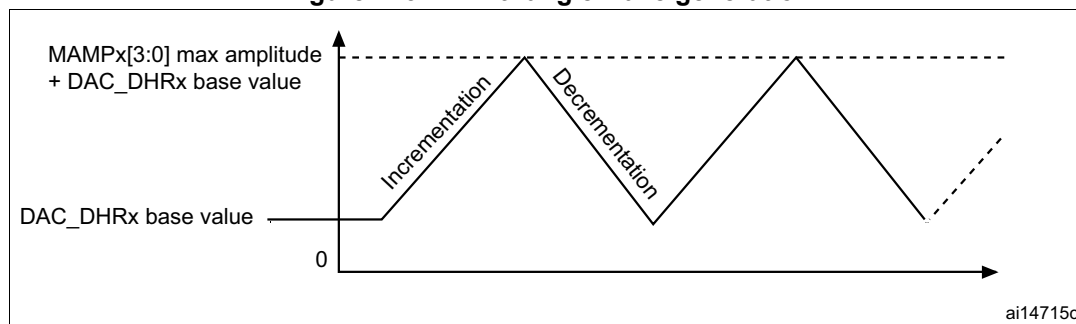
### 22.4.9 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to 10". The amplitude is configured through the MAMPx[3:0] bits in the DAC\_CR register. An internal triangle counter is incremented three APB1 clock cycles after each trigger event. The value of this counter is then added to the DAC\_DHRx register without overflow and the sum is transferred into the DAC\_DORx register. The triangle counter is incremented as long as it is less than the

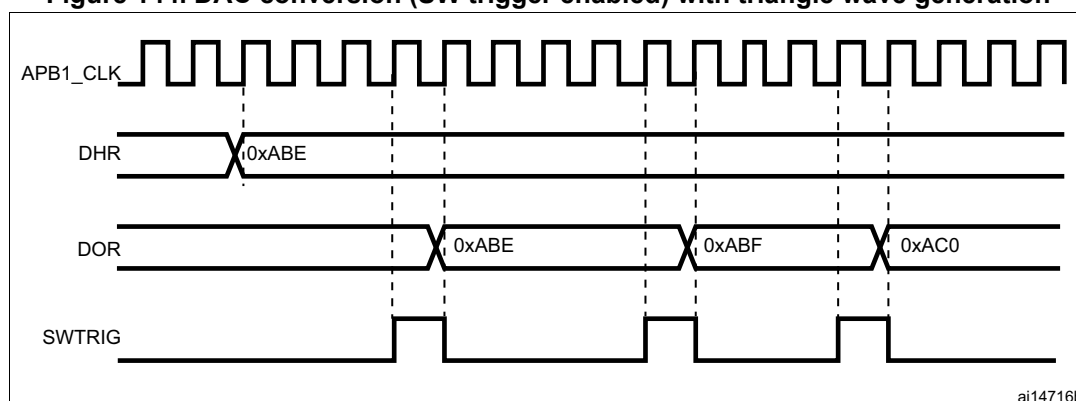
maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

**Figure 143. DAC triangle wave generation**



**Figure 144. DAC conversion (SW trigger enabled) with triangle wave generation**



*Note:* The DAC trigger must be enabled for triangle wave generation by setting the TENx bit in the DAC\_CR register.

The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

### 22.4.10 DAC channel modes

Each DAC channel can be configured in Normal mode or Sample and Hold mode. The output buffer can be enabled to allow a high drive capability. Before enabling output buffer, the voltage offset needs to be calibrated. This calibration is performed at the factory (loaded after reset) and can be adjusted by software during application operation.

#### Normal mode

In Normal mode, there are four combinations, by changing the buffer state and by changing the DAC\_OUTx pin interconnections.

To enable the output buffer, the MODEx[2:0] bits in DAC\_MCR register should be:

- 000: DAC is connected to the external pin
- 001: DAC is connected to external pin and to on-chip peripherals

To disable the output buffer, the MODEx[2:0] bits in DAC\_MCR register should be:

- 010: DAC is connected to the external pin
- 011: DAC is connected to on-chip peripherals

### Sample and Hold mode

In sample and Hold mode, the DAC core converts data on a triggered conversion, then, holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A new stabilization period, which value depends on the buffer state, is required before each new conversion.

In this mode, the DAC core and all corresponding logic and registers are driven by the low-speed clock (LSI) in addition to the APB1 clock, allowing to use the DAC channels in deep low power modes such as Stop mode.

The sample/hold mode operations can be divided into 3 phases:

1. Sample phase: the sample/hold element is charged to the desired voltage. The charging time depends on capacitor value (internal or external, selected by the user). The sampling time is configured with the TSAMPLEx[9:0] bits in DAC\_SHSRx register. During the write of the TSAMPLEx[9:0] bits; the BWSTx bit in DAC\_SR register is set to 1 to synchronize between both clocks domains (APB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation
2. Hold phase: the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the THOLDx[9:0] bits in DAC\_SHHR register
3. Refresh phase: the refresh time is configured with the TREFRESHx[7:0] bits in DAC\_SHRR register

The timings for the three phases above are in units of LSI clocks. As an example, to configure a sample time of 350  $\mu$ s, a hold time of 2 ms and a refresh time of 100  $\mu$ s assuming LSI ~32 KHz is selected:

12 cycles are required for sample phase: TSAMPLEx[9:0] = 11,

62 cycles are required for hold phase: THOLDx[9:0] = 62,

and 4 cycles are required for refresh period: TREFRESHx[7:0] = 4.

In this example, the power consumption is reduced by almost a factor of 15 versus Normal modes.

The formulas to compute the right sample and refresh timings are described in the table below, the Hold time depends on the leakage current.

**Table 132. Sample and refresh timings**

Buffer State	$t_{\text{SAMP}}^{(1)(2)}$	$t_{\text{REFRESH}}^{(2)(3)}$
Enable	$7 \mu\text{s} + (10 \cdot R_{\text{BON}} \cdot C_{\text{SH}})$	$7 \mu\text{s} + (R_{\text{BON}} \cdot C_{\text{SH}}) \cdot \ln(2 \cdot N_{\text{LSB}})$
Disable	$3 \mu\text{s} + (10 \cdot R_{\text{BOFF}} \cdot C_{\text{SH}})$	$3 \mu\text{s} + (R_{\text{BOFF}} \cdot C_{\text{SH}}) \cdot \ln(2 \cdot N_{\text{LSB}})$

1. In the above formula the settling to the desired code value with  $\frac{1}{2}$  LSB or accuracy requires 10 constant time for 12 bits resolution. For 8 bits resolution, the settling time is 7 constant time.

2.  $C_{\text{SH}}$  is the capacitor in Sample and Hold mode.

- The tolerated voltage drop during the hold phase "Vd" is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with 1/2 LSB error accuracy requires  $\ln(2 \cdot N_{LSB})$  constant time of the DAC.

**Example of the sample and refresh time calculation with output buffer on**

The values used in the example below are provided as indication only. Please refer to the product datasheet for product data.

$C_{SH} = 100 \text{ nF}$

$V_{DDA} = 3.0 \text{ V}$

Sampling phase:

$t_{SAMP} = 7 \mu\text{s} + (10 \cdot 2000 \cdot 100 \cdot 10^{-9}) = 2.007 \text{ ms}$   
 (where  $R_{BON} = 2 \text{ k}\Omega$ )

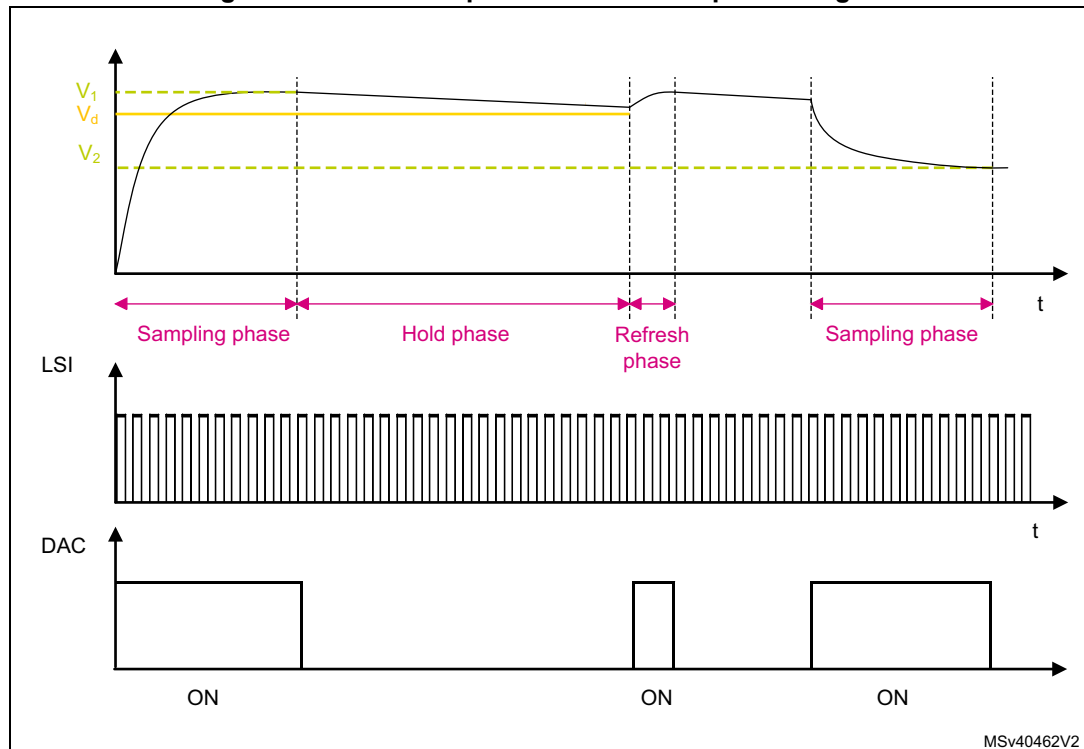
Refresh phase:

$t_{REFRESH} = 7 \mu\text{s} + (2000 \cdot 100 \cdot 10^{-9}) \cdot \ln(2 \cdot 10) = 606.1 \mu\text{s}$   
 (where  $N_{LSB} = 10$  (10 LSB drop during the hold phase))

Hold phase:

$D_V = i_{leak} \cdot t_{hold} / C_{SH} = 0.0073 \text{ V}$  (10 LSB of 12bit at 3 V)  
 $i_{leak} = 150 \text{ nA}$  (worst case on the IO leakage on all the temperature range)  
 $t_{hold} = 0.0073 \cdot 100 \cdot 10^{-9} / (150 \cdot 10^{-9}) = 4.867 \text{ ms}$

**Figure 145. DAC Sample and Hold mode phase diagram**



Like in Normal mode, the Sample and Hold mode has different configurations.

To enable the output buffer, the MODEx[2:0] bits in DAC\_MCR register should be:

- 100: DAC is connected to the external pin
- 101: DAC is connected to external pin and to on chip peripherals

To disabled the output buffer, The MODEx[2:0] bits in DAC\_MCR register should be:

- 110: DAC is connected to external pin and to on chip peripherals
- 111: DAC is connected to on chip peripherals

When MODEx[2:0] bits in DAC\_MCR register is equal to 111. An internal capacitor, C<sub>Lint</sub> will hold the voltage output of the DAC Core and then drive it to on-chip peripherals.

All Sample and Hold phases are interruptible and any change in DAC\_DHRx will trigger immediately a new sample phase.

**Table 133. Channel output modes summary**

MODEx[2:0]			Mode	Buffer	Output connections
0	0	0	Normal mode	Enabled	Connected to external pin
0	0	1			Connected to external pin and to on chip-peripherals (ex, comparators)
0	1	0		Disabled	Connected to external pin
0	1	1			Connected to on chip peripherals (ex, comparators)
1	0	0	Sample and Hold mode	Enabled	Connected to external pin
1	0	1			Connected to external pin and to on chip peripherals (ex, comparators)
1	1	0		Disabled	Connected to external pin and to on chip peripherals (ex, comparators)
1	1	1			Connected to on chip peripherals (ex, comparators)

### 22.4.11 DAC channel buffer calibration

The transfer function for an N-bit digital-to-analog converter (DAC) is:

$$V_{out} = ((D / 2^{N-1}) \times G \times V_{ref}) + V_{OS}$$

Where V<sub>OUT</sub> is the analog output, D is the digital input, G is the gain, V<sub>ref</sub> is the nominal full-scale voltage, and V<sub>os</sub> is the offset voltage. For an ideal DAC channel, G = 1 and V<sub>os</sub> = 0.

Due to output buffer characteristics, the voltage offset may differ from part-to-part and introduce an absolute offset error on the analog output. To compensate the V<sub>os</sub>, a calibration is required by a trimming technique.

The calibration is only valid when the DAC channelx is operating with buffer enabled (MODEx[2:0] = 000b or 001b or 100b or 101b). if applied in other modes when the buffer is off, it has no effect. During the calibration:

- The buffer output will be disconnected from the pin internal/external connections and put in tristate mode (HiZ),
- The buffer will act as a comparator, to sense the middle-code value 0x800 and compare it to VREF+/2 signal through an internal bridge, then toggle its output signal to 0 or 1 depending on the comparison result (CAL\_FLAGx bit)

Two calibration techniques are provided:

- Factory trimming (always enabled)  
The DAC buffer offset is factory trimmed. The default value of OTRIMx[4:0] bits in DAC\_CCR register is the factory trimming value and it is loaded once DAC digital interface is reset.
- User trimming  
The user trimming can be done when the operating conditions differs from nominal factory trimming conditions and in particular when  $V_{DD}/V_{DDA}$  voltage, temperature, VREF+ values change and can be done at any point during application by software.

*Note:* Refer to the datasheet for more details of the Nominal factory trimming conditions

In addition, when  $V_{DD}/V_{DDA}$  is removed (example the device enters in STANDBY or VBAT modes) the calibration is required.

The steps to perform a user trimming calibration are as below:

1. If the DAC channel is active, Write 0 to ENx bit in DAC\_CR to disable the channel.
2. Select a mode where the buffer is enabled, by writing to DAC\_MCR register, MODEx[2:0] = 000b or 001b or 100b or 101b,
3. Start the DAC channelx calibration, by setting the CENx bit in DAC\_CR register to 1,
4. Apply a trimming algorithm:
  - a) Write a code into OTRIMx[4:0] bits, starting by 00000b.
  - b) Wait for  $t_{TRIM}$  delay.
  - c) Check if CAL\_FLAGx bit in DAC\_SR is set to 1.
  - d) if CAL\_FLAGx is set to 1 the trimming code OTRIMx[4:0] is found and will be used during operation to compensate the output value, else increment OTRIMx[4:0] and repeat sub-steps from (a) to (d) again.

The software algorithm may use either a successive approximation or dichotomy techniques to compute and set the content of OTRIMx[4:0] bits in a faster way,

The commutation/toggle of CAL\_FLAGx bit indicates that the offset is correctly compensated and the corresponding trim code must be kept in the OTRIMx[4:0] bits in DAC\_CCR register.

*Note:* A  $t_{TRIM}$  delay must be respected between the write to the OTRIMx[4:0] bits and the read of the CAL\_FLAGx bit in DAC\_SR register in order to get a correct value. This parameter is specified into datasheet electrical characteristics section.

*If the  $V_{DD}/V_{DDA}$ , VREF+ and temperature conditions will not change during the device operation while it enters more often in standby and VBAT mode, the software may store the OTRIMx[4:0] bits found in the first user calibration in the flash or in back-up registers. then to load/write them directly when the device power is back again thus avoiding to wait for a new calibration time.*

*When CENx bit is set, it is not allowed to set ENx bit.*

### 22.4.12 Dual DAC channel conversion (if available)

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time. For the wave generation, no accesses to DHRxxxD registers are required. As a result, two output channels can be used either independently or simultaneously.

11 possible conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DHRx registers if needed.

All modes are described in the paragraphs below.

#### Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits.
3. Load the dual DAC channel data into the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD).

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC\_DOR1 (three APB1 clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC\_DOR2 (three APB1 clock cycles later).

#### Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DHR12RD, DHR12LD or DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three APB1 clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three APB1 clock cycles later). Then the LFSR2 counter is updated.



### Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three APB1 clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three APB1 clock cycles later). Then the LFSR2 counter is updated.

### Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three APB1 clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

### Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is

transferred into DAC\_DOR1 (three APB1 clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

### Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD)

In this configuration, one APB1 clock cycle later, the DHR1 and DHR2 registers are transferred into DAC\_DOR1 and DAC\_DOR2, respectively.

### Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Load the dual DAC channel data to the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD)

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC\_DOR1 and DAC\_DOR2, respectively (after three APB1 clock cycles).

### Simultaneous trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits
- Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD)

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three APB1 clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three APB1 clock cycles later). The LFSR2 counter is then updated.

### Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD)

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three APB1 clock cycles later). The LFSR1 counter is then updated.

At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three APB1 clock cycles later). The LFSR2 counter is then updated.

### Simultaneous trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value using the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD)

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three APB1 clock cycles later). The DAC channel1 triangle counter is then updated.

At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

### Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD)

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three APB clock cycles later). Then the DAC channel1 triangle counter is updated.

At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three APB1 clock cycles later). Then the DAC channel2 triangle counter is updated.

## 22.5 DAC low-power modes

Table 134. Effect of low-power modes on DAC

Mode	Description
Sleep	No effect, DAC used with DMA
Low-power run	No effect.
Low-power sleep	No effect. DAC used with DMA.
Stop 0 / Stop 1	DAC remains active with a static value, if Sample and Hold mode is selected using LSI clock
Stop 2	The DAC registers content is kept. The DAC must be disabled before entering Stop 2.
Standby	The DAC peripheral is powered down and must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	

## 22.6 DAC interrupts

Table 135. DAC interrupts

Interrupt event	Event flag	Enable control bit
DMA underrun	DMAUDRx	DMAUDRIEx

## 22.7 DAC registers

Refer to [Section 1 on page 75](#) for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

### 22.7.1 DAC control register (DAC\_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CEN2	DMAU DRIE2	DMAE N2	MAMP2[3:0]				WAVE2[1:0]		TSEL22	TSEL21	TSEL20	TEN2	Res.	EN2
	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
HFSEL	CEN1	DMAU DRIE1	DMAE N1	MAMP1[3:0]				WAVE1[1:0]		TSEL12	TSEL11	TSEL10	TEN1	Res.	EN1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

- Bit 31 Reserved, must be kept at reset value.
- Bit 30 **CEN2**: DAC Channel 2 calibration enable  
This bit is set and cleared by software to enable/disable DAC channel 2 calibration, it can be written only if EN2 bit is set to 0 into DAC\_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.  
0: DAC channel 2 in Normal operating mode  
1: DAC channel 2 in calibration mode  
*Note: This bit is available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).*
- Bit 29 **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable  
This bit is set and cleared by software.  
0: DAC channel2 DMA underrun interrupt disabled  
1: DAC channel2 DMA underrun interrupt enabled  
*Note: This bit is available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).*
- Bit 28 **DMAEN2**: DAC channel2 DMA enable  
This bit is set and cleared by software.  
0: DAC channel2 DMA mode disabled  
1: DAC channel2 DMA mode enabled  
*Note: This bit is available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).*
- Bits 27:24 **MAMP2[3:0]**: DAC channel2 mask/amplitude selector  
These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.  
0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1  
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3  
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7  
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15  
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31  
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63  
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127  
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255  
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511  
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023  
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047  
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095  
*Note: These bits are available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).*
- Bits 23:22 **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable  
These bits are set/reset by software.  
00: wave generation disabled  
01: Noise wave generation enabled  
1x: Triangle wave generation enabled  
*Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)  
These bits are available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).*

Bits 21:19 **TSEL2[2:0]**: DAC channel2 trigger selection

These bits select the external event used to trigger DAC channel2

Refer to the trigger selection tables in [Section 22.4.6: DAC trigger selection](#) for the details on trigger configuration and mapping.

*Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).*

*These bits are available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).*

Bit 18 **TEN2**: DAC channel2 trigger enable

This bit is set and cleared by software to enable/disable DAC channel2 trigger

0: DAC channel2 trigger disabled and data written into the DAC\_DHR2 register are transferred one APB1 clock cycle later to the DAC\_DOR2 register

1: DAC channel2 trigger enabled and data from the DAC\_DHR2 register are transferred three APB1 clock cycles later to the DAC\_DOR2 register

*Note: When software trigger is selected, the transfer from the DAC\_DHR2 register to the DAC\_DOR2 register takes only one APB1 clock cycle.*

*These bits are available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).*

Bit 17 Reserved, must be kept at reset value.

Bit 16 **EN2**: DAC channel2 enable

This bit is set and cleared by software to enable/disable DAC channel2.

0: DAC channel2 disabled

1: DAC channel2 enabled

*Note: These bits are available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).*

Bit 15 **HFSEL**: High frequency interface mode enable

This bit is set and cleared by software to enable/disable DAC interface high speed mode

This bit need to be set when APB1 clock frequency is higher than 80 MHz.

0: High frequency interface mode disabled

1: High frequency interface mode enabled

Bit 14 **CEN1**: DAC Channel 1 calibration enable

This bit is set and cleared by software to enable/disable DAC channel 1 calibration, it can be written only if bit EN1=0 into DAC\_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel 1 in Normal operating mode

1: DAC channel 1 in calibration mode

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.

0: DAC channel1 DMA Underrun Interrupt disabled

1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.

0: DAC channel1 DMA mode disabled

1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1  
 0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3  
 0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7  
 0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15  
 0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31  
 0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63  
 0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127  
 0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255  
 1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511  
 1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023  
 1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047  
 ≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

These bits are set and cleared by software.

00: wave generation disabled  
 01: Noise wave generation enabled  
 1x: Triangle wave generation enabled  
 Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 5:3 **TSEL1[2:0]**: DAC channel1 trigger selection

These bits select the external event used to trigger DAC channel1

Refer to the trigger selection tables in [Section 22.4.6: DAC trigger selection](#) for the details on trigger configuration and mapping.

*Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).*

Bit 2 **TEN1**: DAC channel1 trigger enable

This bit is set and cleared by software to enable/disable DAC channel1 trigger.

0: DAC channel1 trigger disabled and data written into the DAC\_DHR1 register are transferred one APB1 clock cycle later to the DAC\_DOR1 register  
 1: DAC channel1 trigger enabled and data from the DAC\_DHR1 register are transferred three APB1 clock cycles later to the DAC\_DOR1 register

*Note: When software trigger is selected, the transfer from the DAC\_DHR1 register to the DAC\_DOR1 register takes only one APB1 clock cycle.*

Bit 1 Reserved, must be kept at reset value.

Bit 0 **EN1**: DAC channel1 enable

This bit is set and cleared by software to enable/disable DAC channel1.

0: DAC channel1 disabled  
 1: DAC channel1 enabled

## 22.7.2 DAC software trigger register (DAC\_SWTRGR)

Address offset: 0x04

Reset value: 0x0000 0000



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
														w	w

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SWTRIG2**: DAC channel2 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

*Note: This bit is cleared by hardware (one APB1 clock cycle later) once the DAC\_DHR2 register value has been loaded into the DAC\_DOR2 register.*

*This bit is available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).*

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

*Note: This bit is cleared by hardware (one APB1 clock cycle later) once the DAC\_DHR1 register value has been loaded into the DAC\_DOR1 register.*

### 22.7.3 DAC channel1 12-bit right-aligned data holding register (DAC\_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software. They specify 12-bit data for DAC channel1.

### 22.7.4 DAC channel1 12-bit left aligned data holding register (DAC\_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

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

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data  
 These bits are written by software.  
 They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

### 22.7.5 DAC channel1 8-bit right aligned data holding register (DAC\_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data  
 These bits are written by software. They specify 8-bit data for DAC channel1.

### 22.7.6 DAC channel2 12-bit right aligned data holding register (DAC\_DHR12R2)

This register is available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).

Address offset: 0x14

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data

These bits are written by software. They specify 12-bit data for DAC channel2.

### 22.7.7 DAC channel2 12-bit left aligned data holding register (DAC\_DHR12L2)

This register is available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

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

Bits 15:4 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data

These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0 Reserved, must be kept at reset value.

### 22.7.8 DAC channel2 8-bit right-aligned data holding register (DAC\_DHR8R2)

This register is available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data

These bits are written by software which specifies 8-bit data for DAC channel2.

### 22.7.9 Dual DAC 12-bit right-aligned data holding register (DAC\_DHR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 27:16 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data

These bits are written by software which specifies 12-bit data for DAC channel2.

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

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software which specifies 12-bit data for DAC channel1.

### 22.7.10 Dual DAC 12-bit left aligned data holding register (DAC\_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w				

Bits 31:20 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data

These bits are written by software which specifies 12-bit data for DAC channel2.

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

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data

These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

### 22.7.11 Dual DAC 8-bit right aligned data holding register (DAC\_DHR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

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



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:8 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data

These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data

These bits are written by software which specifies 8-bit data for DAC channel1.

### 22.7.12 DAC channel1 data output register (DAC\_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DOR[11:0]**: DAC channel1 data output

These bits are read-only, they contain data output for DAC channel1.

### 22.7.13 DAC channel2 data output register (DAC\_DOR2)

This register is available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DOR[11:0]**: DAC channel2 data output

These bits are read-only, they contain data output for DAC channel2.

### 22.7.14 DAC status register (DAC\_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BWST2	CAL_FLAG2	DMAU DR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BWST1	CAL_FLAG1	DMAU DR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1													

Bit 31 **BWST2**: DAC Channel 2 busy writing sample time flag

This bit is systematically set just after Sample and Hold mode enable. It is set each time the software writes the register DAC\_SHSR2, It is cleared by hardware when the write operation of DAC\_SHSR2 is complete. (It takes about 3 LSI periods of synchronization).

0: There is no write operation of DAC\_SHSR2 ongoing: DAC\_SHSR2 can be written

1: There is a write operation of DAC\_SHSR2 ongoing: DAC\_SHSR2 cannot be written

*Note: This bit is available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).*

Bit 30 **CAL\_FLAG2**: DAC Channel 2 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

*Note: This bit is available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).*

Bit 29 **DMAUDR2**: DAC channel2 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel2

1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate).

*Note: This bit is available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).*

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 Reserved, must be kept at reset value.

Bit 15 **BWST1**: DAC Channel 1 busy writing sample time flag

This bit is systematically set just after Sample and Hold mode enable and is set each time the software writes the register DAC\_SHSR1, It is cleared by hardware when the write operation of DAC\_SHSR1 is complete. (It takes about 3 LSI periods of synchronization).

0: There is no write operation of DAC\_SHSR1 ongoing: DAC\_SHSR1 can be written

1: There is a write operation of DAC\_SHSR1 ongoing: DAC\_SHSR1 cannot be written

Bit 14 **CAL\_FLAG1**: DAC Channel 1 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel1

1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bit 12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bits 10:0 Reserved, must be kept at reset value.

### 22.7.15 DAC calibration control register (DAC\_CCR)

Address offset: 0x38

Reset value: 0x00XX 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4:0]				
											rw	rw	rw	rw	rw

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

Bits 20:16 **OTRIM2[4:0]**: DAC Channel 2 offset trimming value

These bits are available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).

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

Bits 4:0 **OTRIM1[4:0]**: DAC Channel 1 offset trimming value

### 22.7.16 DAC mode control register (DAC\_MCR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE2[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE1[2:0]		
													rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

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

Bits 18:16 **MODE2[2:0]**: DAC Channel 2 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN2=0 and bit CEN2 =0 in the DAC\_CR register). If EN2=1 or CEN2 =1 the write operation is ignored.

They can be set and cleared by software to select the DAC Channel 2 mode:

- DAC Channel 2 in Normal mode
  - 000: DAC Channel 2 is connected to external pin with Buffer enabled
  - 001: DAC Channel 2 is connected to external pin and to on chip peripherals with buffer enabled
  - 010: DAC Channel 2 is connected to external pin with buffer disabled
  - 011: DAC Channel 2 is connected to on chip peripherals with Buffer disabled
- DAC Channel 2 in Sample and Hold mode
  - 100: DAC Channel 2 is connected to external pin with Buffer enabled
  - 101: DAC Channel 2 is connected to external pin and to on chip peripherals with Buffer enabled
  - 110: DAC Channel 2 is connected to external pin and to on chip peripherals with Buffer disabled
  - 111: DAC Channel 2 is connected to on chip peripherals with Buffer disabled

*Note:* This register can be modified only when EN2=0.

Refer to [Section 22.3: DAC implementation](#) for the availability of DAC channel 2.

Bits 15:14 Reserved, must be kept at reset value.

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

Bit 9 Reserved, must be kept at reset value.

Bit 8 Reserved, must be kept at reset value.

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

Bits 2:0 **MODE1[2:0]**: DAC Channel 1 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1=0 and bit CEN1 =0 in the DAC\_CR register). If EN1=1 or CEN1 =1 the write operation is ignored.

They can be set and cleared by software to select the DAC Channel 1 mode:

- DAC Channel 1 in Normal mode
  - 000: DAC Channel 1 is connected to external pin with Buffer enabled
  - 001: DAC Channel 1 is connected to external pin and to on chip peripherals with Buffer enabled
  - 010: DAC Channel 1 is connected to external pin with Buffer disabled
  - 011: DAC Channel 1 is connected to on chip peripherals with Buffer disabled
- DAC Channel 1 in sample & hold mode
  - 100: DAC Channel 1 is connected to external pin with Buffer enabled
  - 101: DAC Channel 1 is connected to external pin and to on chip peripherals with Buffer enabled
  - 110: DAC Channel 1 is connected to external pin and to on chip peripherals with Buffer disabled
  - 111: DAC Channel 1 is connected to on chip peripherals with Buffer disabled

*Note:* This register can be modified only when EN1=0.



### 22.7.17 DAC channel 1 sample and hold sample time register (DAC\_SHSR1)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 9:0 **TSAMPLE1[9:0]**: DAC Channel 1 sample Time (only valid in Sample and Hold mode)

These bits can be written when the DAC channel1 is disabled or also during normal operation. in the latter case, the write can be done only when BWSTx of DAC\_SCR register is low, If BWSTx=1, the write operation is ignored.

*Note:* It represents the number of LSI clocks to perform a sample phase. Sampling time = (TSAMPLE1[9:0] + 1) x LSI clock period.

### 22.7.18 DAC channel 2 sample and hold sample time register (DAC\_SHSR2)

This register is available only on dual-channel DACs. Refer to [Section 22.3: DAC implementation](#).

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 9:0 **TSAMPLE2[9:0]**: DAC Channel 2 sample Time (only valid in Sample and Hold mode)

These bits can be written when the DAC channel2 is disabled or also during normal operation. in the latter case, the write can be done only when BWSTx of DAC\_SR register is low, if BWSTx=1, the write operation is ignored.

*Note:* It represents the number of LSI clocks to perform a sample phase. Sampling time = (TSAMPLE1[9:0] + 1) x LSI clock period.

### 22.7.19 DAC sample and hold time register (DAC\_SHHR)

Address offset: 0x48

Reset value: 0x0001 0001



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **THOLD2[9:0]**: DAC Channel 2 hold time (only valid in Sample and Hold mode).  
 Hold time= (THOLD[9:0]) x LSI clock period

*Note: This register can be modified only when EN2=0.*

*These bits are available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.*

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **THOLD1[9:0]**: DAC Channel 1 hold Time (only valid in Sample and Hold mode)  
 Hold time= (THOLD[9:0]) x LSI clock period

*Note: This register can be modified only when EN2=0.*

*Note: These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit ENx=0 and bit CEN2x=0 in the DAC\_CR register). If ENx=1 or CENx=1 the write operation is ignored.*

### 22.7.20 DAC sample and hold refresh time register (DAC\_SHRR)

Address offset: 0x4C

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:16 **TREFRESH2[7:0]**: DAC Channel 2 refresh Time (only valid in Sample and Hold mode)  
 Refresh time= (TREFRESH[7:0]) x LSI clock period

*Note: This register can be modified only when EN2=0.*

*These bits are available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.*

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

Bits 7:0 **TREFRESH1[7:0]**: DAC Channel 1 refresh Time (only valid in Sample and Hold mode)  
 Refresh time= (TREFRESH[7:0]) x LSI clock period

*Note: This register can be modified only when EN2=0.*

*Note: These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit ENx=0 and bit CEN2x=0 in the DAC\_CR register). If ENx=1 or CENx=1 the write operation is ignored.*

22.7.21 DAC register map

Table 136 summarizes the DAC registers.

Table 136. DAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	DAC_CR	Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]			WAVE2[2:0]			TSEL22	TSEL21	TSEL20	TEN2	Res.	EN2	HFSEL	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]			WAVE1[2:0]			TSEL12	TSEL11	TSEL10	TEN1	Res.	EN1	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x04	DAC_SWTRGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																															0	0	
0x08	DAC_DHR12R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[11:0]												
	Reset value																																	0
0x0C	DAC_DHR12L1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[11:0]												
	Reset value																																	
0x10	DAC_DHR8R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																	0
0x14	DAC_DHR12R2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x18	DAC_DHR12L2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x1C	DAC_DHR8R2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																	0
0x20	DAC_DHR12RD	Res.	Res.	Res.	Res.	DACC2DHR[11:0]											Res.	Res.	Res.	Res.	DACC1DHR[11:0]													
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	DAC_DHR12LD	DACC2DHR[11:0]											Res.	Res.	Res.	Res.	DACC1DHR[11:0]																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	DAC_DHR8RD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x2C	DAC_DOR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x30	DAC_DOR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x34	DAC_SR	BWST2	CAL_FLAG2	DMAUDR2	Res.	Res.	Res.	Res.	Res.	Res.	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																														



Table 136. DAC 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			
0x38	DAC_CCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OTRIM2[4:0]				Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OTRIM1[4:0]							
	Reset value												X	X	X	X	X													X	X	X	X	X		
0x3C	DAC_MCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	MODE2 [2:0]			Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	MODE1 [2:0]				
	Reset value															0	0	0														0	0	0		
0x40	DAC_SHSR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TSAMPLE1[9:0]												
	Reset value																								0	0	0	0	0	0	0	0	0	0		
0x44	DAC_SHSR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TSAMPLE2[9:0]												
	Reset value																								0	0	0	0	0	0	0	0	0	0		
0x48	DAC_SHHR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	THOLD2[9:0]				Res	Res	Res	Res	Res	Res	Res	Res	THOLD1[9:0]											
	Reset value																								0	0	0	0	0	0	0	0	0	0	1	
0x4C	DAC_SHRR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TREFRESH2[7:0]				Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TREFRESH1[7:0]										
	Reset value																																	1		

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.

## 23 Digital camera interface (DCMI)

### 23.1 DCMI introduction

The digital camera is a synchronous parallel interface able to receive a high-speed data flow from an external 8-, 10-, 12- or 14-bit CMOS camera module. It supports different data formats: YCbCr4:2:2/RGB565 progressive video and compressed data (JPEG).

This interface is for use with black & white cameras, X24 and X5 cameras, and it is assumed that all preprocessing like resizing is performed in the camera module.

### 23.2 DCMI main features

- 8-, 10-, 12- or 14-bit parallel interface
- Embedded/external line and frame synchronization
- Continuous or snapshot mode
- Crop feature
- Supports the following data formats:
  - 8/10/12/14- bit progressive video: either monochrome or raw bayer
  - YCbCr 4:2:2 progressive video
  - RGB 565 progressive video
  - Compressed data: JPEG

### 23.3 DCMI clocks

The digital camera interface uses two clock domains, DCMI\_PIXCLK and HCLK. The signals generated with DCMI\_PIXCLK are sampled on the rising edge of HCLK once they are stable. An enable signal is generated in the HCLK domain, to indicate that data coming from the camera are stable and can be sampled. The maximum DCMI\_PIXCLK period must be higher than 2.5 HCLK periods.

### 23.4 DCMI functional overview

The digital camera interface is a synchronous parallel interface that can receive high-speed data flows. It consists of up to 14 data lines (D13-D0) and a pixel clock line (DCMI\_PIXCLK). The pixel clock has a programmable polarity, so that data can be captured on either the rising or the falling edge of the pixel clock.

The data are packed into a 32-bit data register (DCMI\_DR) and then transferred through a general-purpose DMA channel. The image buffer is managed by the DMA, not by the camera interface.

The data received from the camera can be organized in lines/frames (raw YUB/RGB/Bayer modes) or can be a sequence of JPEG images. To enable JPEG image reception, the JPEG bit (bit 3 of DCMI\_CR register) must be set.

The data flow is synchronized either by hardware using the optional DCMI\_HSYNC (horizontal synchronization) and DCMI\_VSYNC (vertical synchronization) signals or by synchronization codes embedded in the data flow.

### 23.4.1 DCMI block diagram

Figure 146 shows the DCMI block diagram.

Figure 146. DCMI block diagram

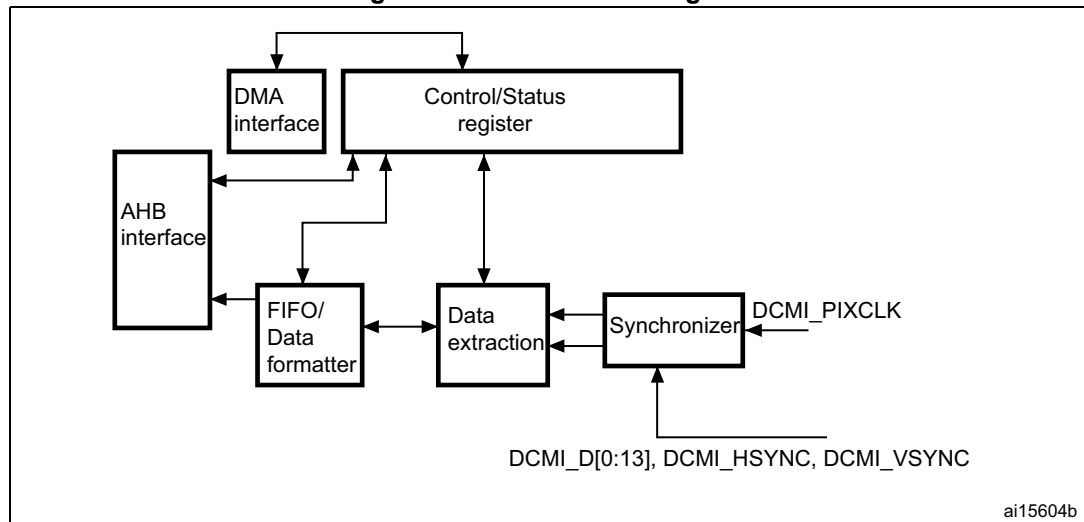
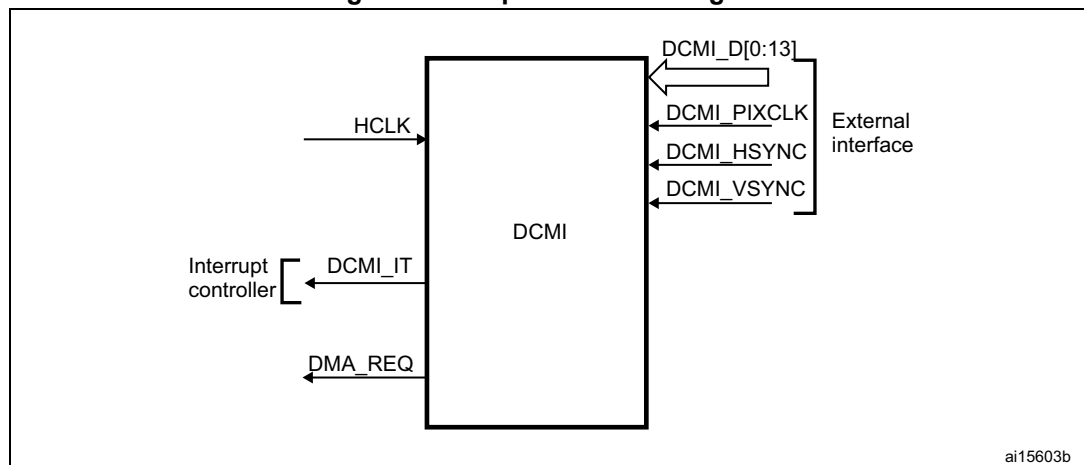


Figure 147. Top-level block diagram



### 23.4.2 DMA interface

The DMA interface is active when the CAPTURE bit in the DCMI\_CR register is set. A DMA request is generated each time the camera interface receives a complete 32-bit data block in its register.

### 23.4.3 DCMI physical interface

The interface is composed of 11/13/15/17 inputs. Only the Slave mode is supported.

The camera interface can capture 8-bit, 10-bit, 12-bit or 14-bit data depending on the EDM[1:0] bits in the DCMI\_CR register. If less than 14 bits are used, the unused input pins must be connected to ground.

Table 137 shows the DCMI pins.

**Table 137. DCMI external signals**

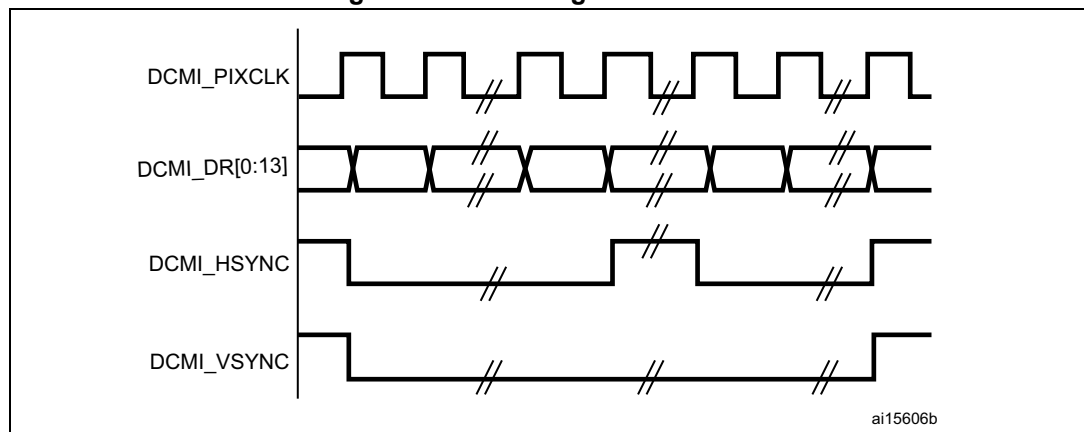
Signal name		Signal type	Signal description
8 bits	DCMI_D[0..7]	Digital inputs	DCMI data
10 bits	DCMI_D[0..9]		
12 bits	DCMI_D[0..11]		
14 bits	DCMI_D[0..13]		
DCMI_PIXCLK		Digital input	Pixel clock
DCMI_HSYNC		Digital input	Horizontal synchronization / Data valid
DCMI_VSYNC		Digital input	Vertical synchronization

The data are synchronous with DCMI\_PIXCLK and change on the rising/falling edge of the pixel clock depending on the polarity.

The DCMI\_HSYNC signal indicates the start/end of a line.

The DCMI\_VSYNC signal indicates the start/end of a frame

**Figure 148. DCMI signal waveforms**



1. The capture edge of DCMI\_PIXCLK is the falling edge, the active state of DCMI\_HSYNC and DCMI\_VSYNC is 1.
2. DCMI\_HSYNC and DCMI\_VSYNC can change states at the same time.

**8-bit data**

When EDM[1:0] in DCMI\_CR are programmed to “00” the interface captures 8 LSBs at its input (DCMI\_D[0:7]) and stores them as 8-bit data. The DCMI\_D[13:8] inputs are ignored. In this case, to capture a 32-bit word, the camera interface takes four pixel clock cycles.

The first captured data byte is placed in the LSB position in the 32-bit word and the 4<sup>th</sup> captured data byte is placed in the MSB position in the 32-bit word. Table 138 gives an example of the positioning of captured data bytes in two 32-bit words.



**Table 138. Positioning of captured data bytes in 32-bit words (8-bit width)**

Byte address	31:24	23:16	15:8	7:0
0	$D_{n+3}[7:0]$	$D_{n+2}[7:0]$	$D_{n+1}[7:0]$	$D_n[7:0]$
4	$D_{n+7}[7:0]$	$D_{n+6}[7:0]$	$D_{n+5}[7:0]$	$D_{n+4}[7:0]$

**10-bit data**

When EDM[1:0] in DCMI\_CR are programmed to “01”, the camera interface captures 10-bit data at its input DCMI\_D[0..9] and stores them as the 10 least significant bits of a 16-bit word. The remaining most significant bits in the DCMI\_DR register (bits 11 to 15) are cleared to zero. So, in this case, a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2<sup>nd</sup> captured data are placed in the MSB position in the 32-bit word as shown in [Table 139](#).

**Table 139. Positioning of captured data bytes in 32-bit words (10-bit width)**

Byte address	31:26	25:16	15:10	9:0
0	0	$D_{n+1}[9:0]$	0	$D_n[9:0]$
4	0	$D_{n+3}[9:0]$	0	$D_{n+2}[9:0]$

**12-bit data**

When EDM[1:0] in DCMI\_CR are programmed to “10”, the camera interface captures the 12-bit data at its input DCMI\_D[0..11] and stores them as the 12 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2<sup>nd</sup> captured data are placed in the MSB position in the 32-bit word as shown in [Table 140](#).

**Table 140. Positioning of captured data bytes in 32-bit words (12-bit width)**

Byte address	31:28	27:16	15:12	11:0
0	0	$D_{n+1}[11:0]$	0	$D_n[11:0]$
4	0	$D_{n+3}[11:0]$	0	$D_{n+2}[11:0]$

**14-bit data**

When EDM[1:0] in DCMI\_CR are programmed to “11”, the camera interface captures the 14-bit data at its input DCMI\_D[0..13] and stores them as the 14 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2<sup>nd</sup> captured data are placed in the MSB position in the 32-bit word as shown in [Table 141](#).

**Table 141. Positioning of captured data bytes in 32-bit words (14-bit width)**

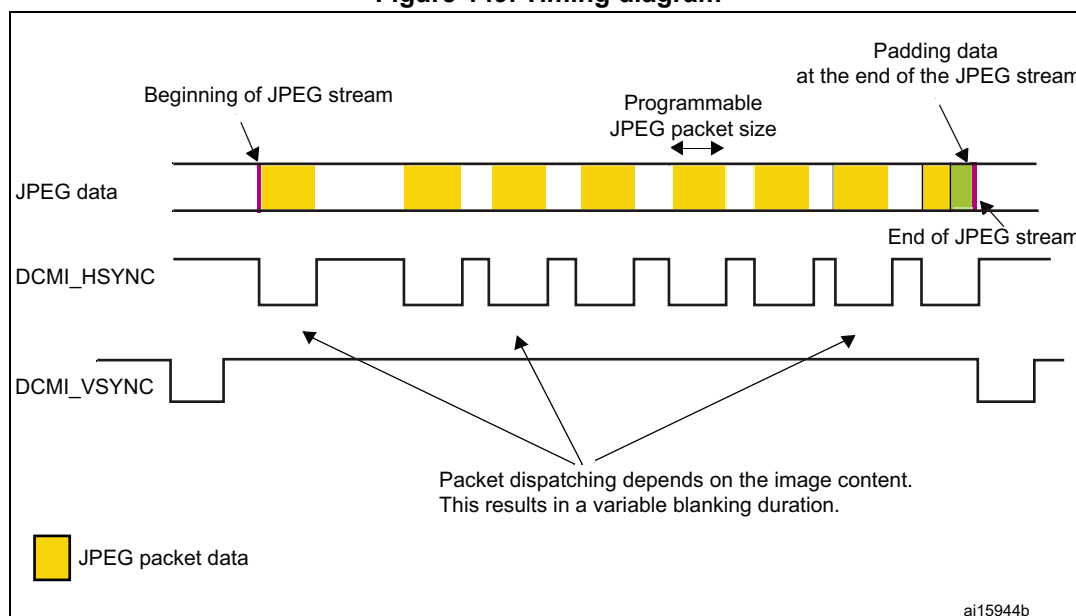
Byte address	31:30	29:16	15:14	13:0
0	0	$D_{n+1}[13:0]$	0	$D_n[13:0]$
4	0	$D_{n+3}[13:0]$	0	$D_{n+2}[13:0]$

### 23.4.4 Synchronization

The digital camera interface supports embedded or hardware (DCMI\_HSYNC and DCMI\_VSYNC) synchronization. When embedded synchronization is used, it is up to the digital camera module to make sure that the 0x00 and 0xFF values are used ONLY for synchronization (not in data). Embedded synchronization codes are supported only for the 8-bit parallel data interface width (that is, in the DCMI\_CR register, the EDM[1:0] bits should be cleared to “00”).

For compressed data, the DCMI supports only the hardware synchronization mode. In this case, DCMI\_VSYNC is used as a start/end of the image, and DCMI\_HSYNC is used as a Data Valid signal. *Figure 149* shows the corresponding timing diagram.

**Figure 149. Timing diagram**



#### Hardware synchronization mode

In hardware synchronization mode, the two synchronization signals (DCMI\_HSYNC/DCMI\_VSYNC) are used.

Depending on the camera module/mode, data may be transmitted during horizontal/vertical synchronization periods. The DCMI\_HSYNC/DCMI\_VSYNC signals act like blanking signals since all the data received during DCMI\_HSYNC/DCMI\_VSYNC active periods are ignored.

In order to correctly transfer images into the DMA/RAM buffer, data transfer is synchronized with the DCMI\_VSYNC signal. When the hardware synchronization mode is selected, and

capture is enabled (CAPTURE bit set in DCMI\_CR), data transfer is synchronized with the deactivation of the DCMI\_VSYNC signal (next start of frame).

Transfer can then be continuous, with successive frames transferred by DMA to successive buffers or the same/circular buffer. To allow the DMA management of successive frames, a VSIF (Vertical synchronization interrupt flag) is activated at the end of each frame.

### Embedded data synchronization mode

In this synchronization mode, the data flow is synchronized using 32-bit codes embedded in the data flow. These codes use the 0x00/0xFF values that are *not* used in data anymore. There are 4 types of codes, all with a 0xFF0000XY format. The embedded synchronization codes are supported only in 8-bit parallel data width capture (in the DCMI\_CR register, the EDM[1:0] bits should be programmed to "00"). For other data widths, this mode generates unpredictable results and must not be used.

*Note: Camera modules can have 8 such codes (in interleaved mode). For this reason, the interleaved mode is not supported by the camera interface (otherwise, every other half-frame would be discarded).*

- Mode 2

Four embedded codes signal the following events

- Frame start (FS)
- Frame end (FE)
- Line start (LS)
- Line end (LE)

The XY values in the 0xFF0000XY format of the four codes are programmable (see [Section 23.7.7: DCMI embedded synchronization code register \(DCMI\\_ESCR\)](#)).

A 0xFF value programmed as a "frame end" means that all the unused codes are interpreted as valid frame end codes.

In this mode, once the camera interface has been enabled, the frame capture starts after the first occurrence of the frame end (FE) code followed by a frame start (FS) code.

- Mode 1

An alternative coding is the camera mode 1. This mode is ITU656 compatible.

The codes signal another set of events:

- SAV (active line) - line start
- EAV (active line) - line end
- SAV (blanking) - end of line during interframe blanking period
- EAV (blanking) - end of line during interframe blanking period

This mode can be supported by programming the following codes:

- FS ≤ 0xFF
- FE ≤ 0xFF
- LS ≤ SAV (active)
- LE ≤ EAV (active)

An embedded unmask code is also implemented for frame/line start and frame/line end codes. Using it, it is possible to compare only the selected unmasked bits with the programmed code. You can therefore select a bit to compare in the embedded code and

detect a frame/line start or frame/line end. This means that there can be different codes for the frame/line start and frame/line end with the unmasked bit position remaining the same.

**Example**

FS = 0xA5

Unmask code for FS = 0x10

In this case the frame start code is embedded in the bit 4 of the frame start code.

**23.4.5 Capture modes**

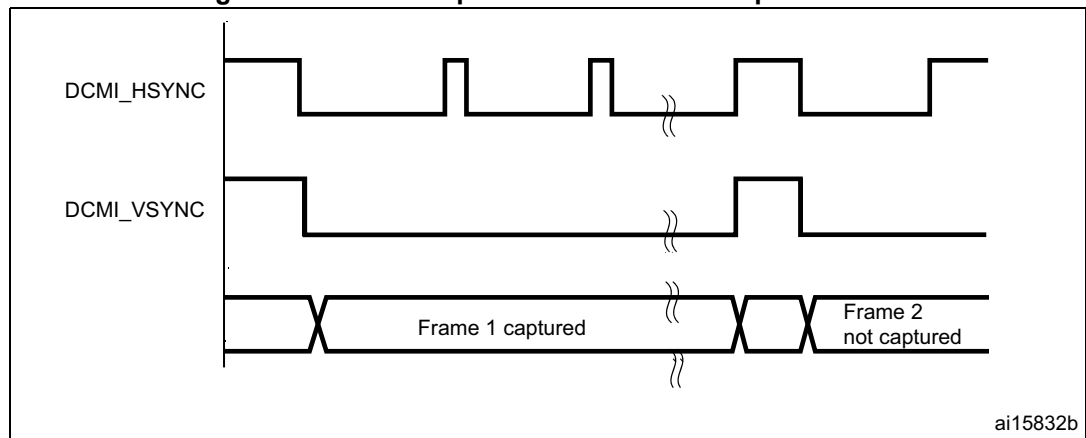
This interface supports two types of capture: snapshot (single frame) and continuous grab.

**Snapshot mode (single frame)**

In this mode, a single frame is captured (CM = '1' in the DCMI\_CR register). After the CAPTURE bit is set in DCMI\_CR, the interface waits for the detection of a start of frame before sampling the data. The camera interface is automatically disabled (CAPTURE bit cleared in DCMI\_CR) after receiving the first complete frame. An interrupt is generated (IT\_FRAME) if it is enabled.

In case of an overrun, the frame is lost and the CAPTURE bit is cleared.

**Figure 150. Frame capture waveforms in snapshot mode**

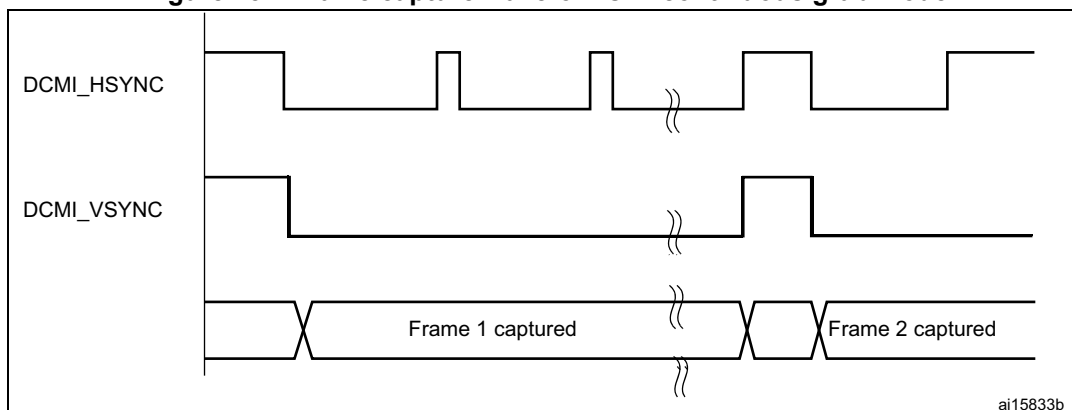


1. Here, the active state of DCMI\_HSYNC and DCMI\_VSYNC is 1.
2. DCMI\_HSYNC and DCMI\_VSYNC can change states at the same time.

**Continuous grab mode**

In this mode (CM bit = '0' in DCMI\_CR), once the CAPTURE bit has been set in DCMI\_CR, the grabbing process starts on the next DCMI\_VSYNC or embedded frame start depending on the mode. The process continues until the CAPTURE bit is cleared in DCMI\_CR. Once the CAPTURE bit has been cleared, the grabbing process continues until the end of the current frame.

Figure 151. Frame capture waveforms in continuous grab mode



1. Here, the active state of DCMI\_HSYNC and DCMI\_VSYNC is 1.
2. DCMI\_HSYNC and DCMI\_VSYNC can change states at the same time.

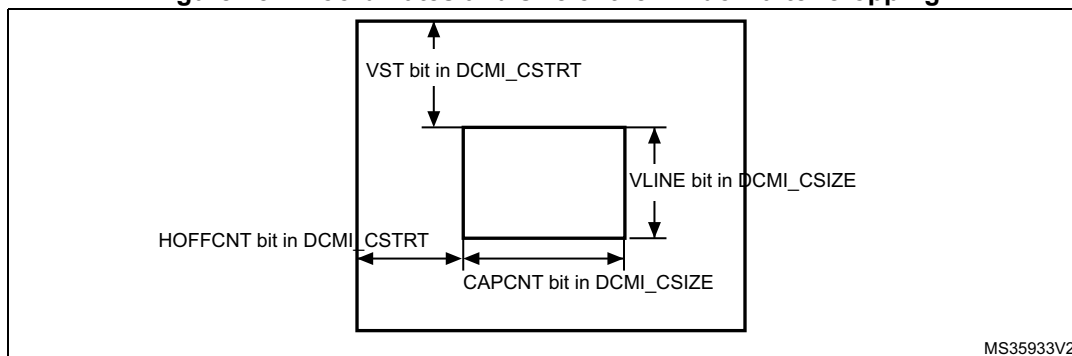
In continuous grab mode, you can configure the FCRC bits in DCMI\_CR to grab all pictures, every second picture or one out of four pictures to decrease the frame capture rate.

*Note:* In the hardware synchronization mode (ESS = '0' in DCMI\_CR), the IT\_VSYNC interrupt is generated (if enabled) even when CAPTURE = '0' in DCMI\_CR so, to reduce the frame capture rate even further, the IT\_VSYNC interrupt can be used to count the number of frames between 2 captures in conjunction with the Snapshot mode. This is not allowed by embedded data synchronization mode.

### 23.4.6 Crop feature

With the crop feature, the camera interface can select a rectangular window from the received image. The start (upper left corner) coordinates and size (horizontal dimension in number of pixel clocks and vertical dimension in number of lines) are specified using two 32-bit registers (DCMI\_CWSTRT and DCMI\_CWSIZE). The size of the window is specified in number of pixel clocks (horizontal dimension) and in number of lines (vertical dimension).

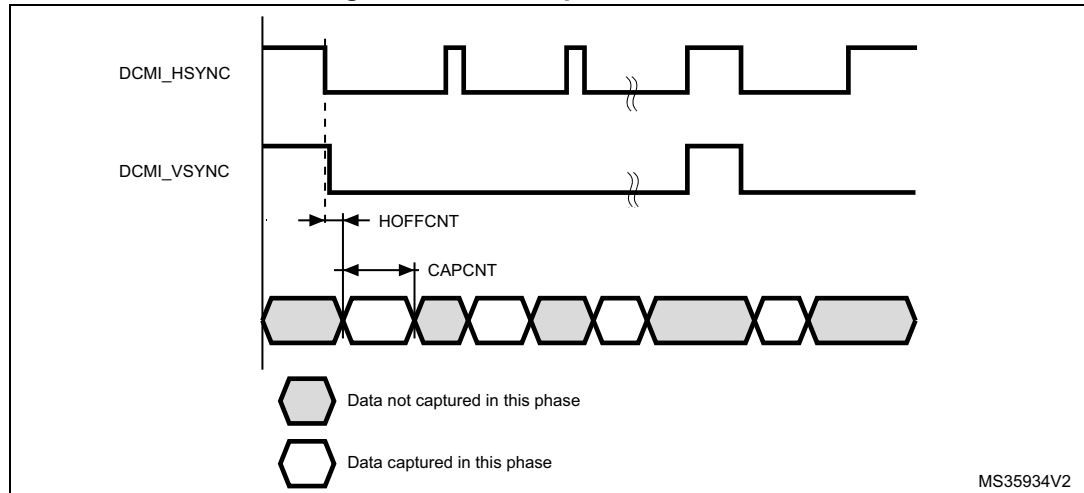
Figure 152. Coordinates and size of the window after cropping



These registers specify the coordinates of the starting point of the capture window as a line number (in the frame, starting from 0) and a number of pixel clocks (on the line, starting from 0), and the size of the window as a line number and a number of pixel clocks. The CAPCNT value can only be a multiple of 4 (two least significant bits are forced to 0) to allow the correct transfer of data through the DMA.

If the DCMI\_VSYNC signal goes active before the number of lines is specified in the DCMI\_CWSIZE register, then the capture stops and an IT\_FRAME interrupt is generated when enabled.

Figure 153. Data capture waveforms



1. Here, the active state of DCMI\_HSYNC and DCMI\_VSYNC is 1.
2. DCMI\_HSYNC and DCMI\_VSYNC can change states at the same time.

### 23.4.7 JPEG format

To allow JPEG image reception, it is necessary to set the JPEG bit in the DCMI\_CR register. JPEG images are not stored as lines and frames, so the DCMI\_VSYNC signal is used to start the capture while DCMI\_HSYNC serves as a data enable signal. The number of bytes in a line may not be a multiple of 4, you should therefore be careful when handling this case since a DMA request is generated each time a complete 32-bit word has been constructed from the captured data. When an end of frame is detected and the 32-bit word to be transferred has not been completely received, the remaining data are padded with '0s' and a DMA request is generated.

The crop feature and embedded synchronization codes cannot be used in the JPEG format.

### 23.4.8 FIFO

#### Input mode

A four-word FIFO is implemented to manage data rate transfers on the AHB. The DCMI features a simple FIFO controller with a read pointer incremented each time the camera interface reads from the AHB, and a write pointer incremented each time the camera interface writes to the FIFO. There is no overrun protection to prevent the data from being overwritten if the AHB interface does not sustain the data transfer rate.

In case of overrun or errors in the synchronization signals, the FIFO is reset and the DCMI interface waits for a new start of frame.

## 23.5 Data format description

### 23.5.1 Data formats

Three types of data are supported:

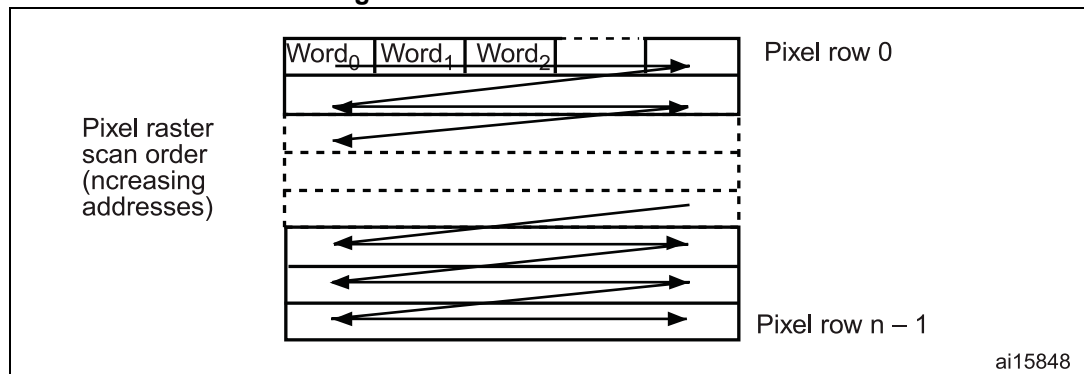
- 8-bit progressive video: either monochrome or raw Bayer format
- YCbCr 4:2:2 progressive video
- RGB565 progressive video. A pixel coded in 16 bits (5 bits for blue, 5 bits for red, 6 bits for green) takes two clock cycles to be transferred.

Compressed data: JPEG

For B&W, YCbCr or RGB data, the maximum input size is 2048 × 2048 pixels. No limit in JPEG compressed mode.

For monochrome, RGB & YCbCr, the frame buffer is stored in raster mode. 32-bit words are used. Only the little endian format is supported.

**Figure 154. Pixel raster scan order**



### 23.5.2 Monochrome format

Characteristics:

- Raster format
- 8 bits per pixel

[Table 142](#) shows how the data are stored.

**Table 142. Data storage in monochrome progressive video format**

Byte address	31:24	23:16	15:8	7:0
0	n + 3	n + 2	n + 1	n
4	n + 7	n + 6	n + 5	n + 4

### 23.5.3 RGB format

Characteristics:

- Raster format
- RGB
- Interleaved: one buffer: R, G & B interleaved: BRGBRGBRG, etc.
- Optimized for display output

The RGB planar format is compatible with standard OS frame buffer display formats. Only 16 BPP (bits per pixel): RGB565 (2 pixels per 32-bit word) is supported. The 24 BPP (palletized format) and grayscale formats are not supported. Pixels are stored in a raster scan order, that is from top to bottom for pixel rows, and from left to right within a pixel row. Pixel components are R (red), G (green) and B (blue). All components have the same spatial resolution (4:4:4 format). A frame is stored in a single part, with the components interleaved on a pixel basis.

[Table 143](#) shows how the data are stored.

**Table 143. Data storage in RGB progressive video format**

Byte address	31:27	26:21	20:16	15:11	10:5	4:0
0	Red n + 1	Green n + 1	Blue n + 1	Red n	Green n	Blue n
4	Red n + 4	Green n + 3	Blue n + 3	Red n + 2	Green n + 2	Blue n + 2

### 23.5.4 YCbCr format

Characteristics:

- Raster format
- YCbCr 4:2:2
- Interleaved: one Buffer: Y, Cb & Cr interleaved: CbYCrYCbYCr, etc.

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). Each component is encoded in 8 bits. Luma and chroma are stored together (interleaved) as shown in [Table 144](#).

**Table 144. Data storage in YCbCr progressive video format**

Byte address	31:24	23:16	15:8	7:0
0	Y n + 1	Cr n	Y n	Cb n
4	Y n + 3	Cr n + 2	Y n + 2	Cb n + 2

### 23.5.5 YCbCr format - Y only

Characteristics:

- Raster format
- YCbCr 4:2:2
- The buffer only contains Y information - monochrome image

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). In this mode, the chroma information is dropped. Only Luma component of each



pixel , encoded in 8 bits, is stored as shown in [Table 145](#).

The result is a monochrome image having the same resolution as the original YCbCr data.

**Table 145. Data storage in YCbCr progressive video format - Y extraction mode**

Byte address	31:24	23:16	15:8	7:0
0	$Y_{n+3}$	$Y_{n+2}$	$Y_{n+1}$	$Y_n$
4	$Y_{n+7}$	$Y_{n+6}$	$Y_{n+5}$	$Y_{n+4}$

### 23.5.6 Half resolution image extraction

This is a modification of the previous reception modes, being applicable to monochrome, RGB or Y extraction modes.

This mode allows to only store a half resolution image. It is selected through OELS and LSM control bits.

## 23.6 DCMI interrupts

Five interrupts are generated. All interrupts are maskable by software. The global interrupt (IT\_DCMI) is the OR of all the individual interrupts. [Table 146](#) gives the list of all interrupts.

**Table 146. DCMI interrupts**

Interrupt name	Interrupt event
IT_LINE	Indicates the end of line
IT_FRAME	Indicates the end of frame capture
IT_OVR	indicates the overrun of data reception
IT_VSYNC	Indicates the synchronization frame
IT_ERR	Indicates the detection of an error in the embedded synchronization frame detection
IT_DCMI	Logic OR of the previous interrupts

## 23.7 DCMI register description

All DCMI registers have to be accessed as 32-bit words, otherwise a bus error occurs.

### 23.7.1 DCMI control register (DCMI\_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OELS	LSM	OEBS	BSM[1:0]	
											r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ENABLE	Res.	Res.	EDM[1:0]		FCRC[1:0]		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
	r/w			r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bit 20 **OELS**: Odd/Even Line Select (Line Select Start)

This bit works in conjunction with LSM field (LSM = 1)

0: Interface captures first line after the frame start, second one being dropped

1: Interface captures second line from the frame start, first one being dropped

Bit 19 **LSM**: Line Select mode

0: Interface captures all received lines

1: Interface captures one line out of two.

Bit 18 **OEBS**: Odd/Even Byte Select (Byte Select Start)

This bit works in conjunction with BSM field (BSM <> 00)

0: Interface captures first data (byte or double byte) from the frame/line start, second one being dropped

1: Interface captures second data (byte or double byte) from the frame/line start, first one being dropped

Bits 17:16 **BSM[1:0]**: Byte Select mode

00: Interface captures all received data

01: Interface captures every other byte from the received data

10: Interface captures one byte out of four

11: Interface captures two bytes out of four

*Note: This mode only work for EDM[1:0]=00. For all other EDM values, this bit field must be programmed to the reset value.*

Bit 15 Reserved, must be kept at reset value.

Bit 14 **ENABLE**: DCMI enable

0: DCMI disabled

1: DCMI enabled

*Note: The DCMI configuration registers should be programmed correctly before enabling this Bit*

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

Bits 11:10 **EDM[1:0]**: Extended data mode

00: Interface captures 8-bit data on every pixel clock

01: Interface captures 10-bit data on every pixel clock

10: Interface captures 12-bit data on every pixel clock

11: Interface captures 14-bit data on every pixel clock

Bits 9:8 **FCRC[1:0]**: Frame capture rate control

These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode. They are ignored in snapshot mode.

00: All frames are captured

01: Every alternate frame captured (50% bandwidth reduction)

10: One frame in 4 frames captured (75% bandwidth reduction)

11: reserved

- Bit 7 **VSPOL**: Vertical synchronization polarity  
This bit indicates the level on the DCMI\_VSYNC pin when the data are not valid on the parallel interface.  
0: DCMI\_VSYNC active low  
1: DCMI\_VSYNC active high
- Bit 6 **HSPOL**: Horizontal synchronization polarity  
This bit indicates the level on the DCMI\_HSYNC pin when the data are not valid on the parallel interface.  
0: DCMI\_HSYNC active low  
1: DCMI\_HSYNC active high
- Bit 5 **PCKPOL**: Pixel clock polarity  
This bit configures the capture edge of the pixel clock  
0: Falling edge active.  
1: Rising edge active.
- Bit 4 **ESS**: Embedded synchronization select  
0: Hardware synchronization data capture (frame/line start/stop) is synchronized with the DCMI\_HSYNC/DCMI\_VSYNC signals.  
1: Embedded synchronization data capture is synchronized with synchronization codes embedded in the data flow.  
*Note: Valid only for 8-bit parallel data. HSPOL/VSPOL are ignored when the ESS bit is set.*  
This bit is disabled in JPEG mode.
- Bit 3 **JPEG**: JPEG format  
0: Uncompressed video format  
1: This bit is used for JPEG data transfers. The DCMI\_HSYNC signal is used as data enable. The crop and embedded synchronization features (ESS bit) cannot be used in this mode.
- Bit 2 **CROP**: Crop feature  
0: The full image is captured. In this case the total number of bytes in an image frame should be a multiple of 4  
1: Only the data inside the window specified by the crop register will be captured. If the size of the crop window exceeds the picture size, then only the picture size is captured.
- Bit 1 **CM**: Capture mode  
0: Continuous grab mode - The received data are transferred into the destination memory through the DMA. The buffer location and mode (linear or circular buffer) is controlled through the system DMA.  
1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame and then transfers a single frame through the DMA. At the end of the frame, the CAPTURE bit is automatically reset.

Bit 0 **CAPTURE**: Capture enable

0: Capture disabled.

1: Capture enabled.

The camera interface waits for the first start of frame, then a DMA request is generated to transfer the received data into the destination memory.

In snapshot mode, the CAPTURE bit is automatically cleared at the end of the 1st frame received.

In continuous grab mode, if the software clears this bit while a capture is ongoing, the bit will be effectively cleared after the frame end.

*Note: The DMA controller and all DCMI configuration registers should be programmed correctly before enabling this bit.*

### 23.7.2 DCMI status register (DCMI\_SR)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

Bit 2 **FNE**: FIFO not empty

This bit gives the status of the FIFO

- 1: FIFO contains valid data
- 0: FIFO empty

Bit 1 **VSYNC**:

This bit gives the state of the DCMI\_VSYNC pin with the correct programmed polarity.

When embedded synchronization codes are used, the meaning of this bit is the following:

- 0: active frame
- 1: synchronization between frames

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI\_CR is set.

Bit 0 **HSYNC**:

This bit gives the state of the DCMI\_HSYNC pin with the correct programmed polarity.

When embedded synchronization codes are used, the meaning of this bit is the following:

- 0: active line
- 1: synchronization between lines

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI\_CR is set.

### 23.7.3 DCMI raw interrupt status register (DCMI\_RIS)

Address offset: 0x08

Reset value: 0x0000 0000

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

DCMI\_RIS gives the raw interrupt status and is accessible in read only. When read, this register returns the status of the corresponding interrupt before masking with the DCMI\_IER register value.

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE\_RIS**: Line raw interrupt status

This bit gets set when the DCMI\_HSYNC signal changes from the inactive state to the active state. It goes high even if the line is not valid.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit in DCMI\_CR is set.

It is cleared by writing a '1' to the LINE\_ISC bit in DCMI\_ICR.

Bit 3 **VSYNC\_RIS**: DCMI\_VSYNC raw interrupt status

This bit is set when the DCMI\_VSYNC signal changes from the inactive state to the active state.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMI\_CR.

It is cleared by writing a '1' to the VSYNC\_ISC bit in DCMI\_ICR.

Bit 2 **ERR\_RIS**: Synchronization error raw interrupt status

0: No synchronization error detected

1: Embedded synchronization characters are not received in the correct order.

This bit is valid only in the embedded synchronization mode. It is cleared by writing a '1' to the ERR\_ISC bit in DCMI\_ICR.

*Note: This bit is available only in embedded synchronization mode.*

Bit 1 **OVR\_RIS**: Overrun raw interrupt status

0: No data buffer overrun occurred

1: A data buffer overrun occurred and the data FIFO is corrupted.

This bit is cleared by writing a '1' to the OVR\_ISC bit in DCMI\_ICR.

Bit 0 **FRAME\_RIS**: Capture complete raw interrupt status

0: No new capture

1: A frame has been captured.

This bit is set when a frame or window has been captured.

In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (e.g. window cropped outside the frame).

This bit is cleared by writing a '1' to the FRAME\_ISC bit in DCMI\_ICR.

### 23.7.4 DCMI interrupt enable register (DCMI\_IER)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE _IE	VSYNC _IE	ERR _IE	OVR _IE	FRAME _IE
											rw	rw	rw	rw	rw

The DCMI\_IER register is used to enable interrupts. When one of the DCMI\_IER bits is set, the corresponding interrupt is enabled. This register is accessible in both read and write.

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE\_IE**: Line interrupt enable

0: No interrupt generation when the line is received

1: An interrupt is generated when a line has been completely received

Bit 3 **VSYNC\_IE**: DCMI\_VSYNC interrupt enable

0: No interrupt generation

1: An interrupt is generated on each DCMI\_VSYNC transition from the inactive to the active state

The active state of the DCMI\_VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR\_IE**: Synchronization error interrupt enable

0: No interrupt generation

1: An interrupt is generated if the embedded synchronization codes are not received in the correct order.

*Note: This bit is available only in embedded synchronization mode.*

Bit 1 **OVR\_IE**: Overrun interrupt enable

0: No interrupt generation

1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received.

Bit 0 **FRAME\_IE**: Capture complete interrupt enable

0: No interrupt generation

1: An interrupt is generated at the end of each received frame/crop window (in crop mode).

### 23.7.5 DCMI masked interrupt status register (DCMI\_MIS)

This DCMI\_MIS register is a read-only register. When read, it returns the current masked status value (depending on the value in DCMI\_IER) of the corresponding interrupt. A bit in this register is set if the corresponding enable bit in DCMI\_IER is set and the corresponding bit in DCMI\_RIS is set.

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

**Bit 4 LINE\_MIS:** Line masked interrupt status

This bit gives the status of the masked line interrupt  
 0: No interrupt generation when the line is received  
 1: An Interrupt is generated when a line has been completely received and the LINE\_IE bit is set in DCMI\_IER.

**Bit 3 VSYNC\_MIS:** VSYNC masked interrupt status

This bit gives the status of the masked VSYNC interrupt  
 0: No interrupt is generated on DCMI\_VSYNC transitions  
 1: An interrupt is generated on each DCMI\_VSYNC transition from the inactive to the active state and the VSYNC\_IE bit is set in DCMI\_IER.  
 The active state of the DCMI\_VSYNC signal is defined by the VSPOL bit.

**Bit 2 ERR\_MIS:** Synchronization error masked interrupt status

This bit gives the status of the masked synchronization error interrupt  
 0: No interrupt is generated on a synchronization error  
 1: An interrupt is generated if the embedded synchronization codes are not received in the correct order and the ERR\_IE bit in DCMI\_IER is set.

*Note: This bit is available only in embedded synchronization mode.*

**Bit 1 OVR\_MIS:** Overflow masked interrupt status

This bit gives the status of the masked overflow interrupt  
 0: No interrupt is generated on overrun  
 1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received and the OVR\_IE bit is set in DCMI\_IER.

**Bit 0 FRAME\_MIS:** Capture complete masked interrupt status

This bit gives the status of the masked capture complete interrupt  
 0: No interrupt is generated after a complete capture  
 1: An interrupt is generated at the end of each received frame/crop window (in crop mode) and the FRAME\_IE bit is set in DCMI\_IER.



### 23.7.6 DCMI interrupt clear register (DCMI\_ICR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE _ISC	VSYNC _ISC	ERR _ISC	OVR _ISC	FRAME _ISC
											w	w	w	w	w

The DCMI\_ICR register is write-only. Writing a '1' into a bit of this register clears the corresponding bit in the DCMI\_RIS and DCMI\_MIS registers. Writing a '0' has no effect.

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE\_ISC**: line interrupt status clear

Writing a '1' into this bit clears LINE\_RIS in the DCMI\_RIS register

Bit 3 **VSYNC\_ISC**: Vertical Synchronization interrupt status clear

Writing a '1' into this bit clears the VSYNC\_RIS bit in DCMI\_RIS

Bit 2 **ERR\_ISC**: Synchronization error interrupt status clear

Writing a '1' into this bit clears the ERR\_RIS bit in DCMI\_RIS

*Note: This bit is available only in embedded synchronization mode.*

Bit 1 **OVR\_ISC**: Overrun interrupt status clear

Writing a '1' into this bit clears the OVR\_RIS bit in DCMI\_RIS

Bit 0 **FRAME\_ISC**: Capture complete interrupt status clear

Writing a '1' into this bit clears the FRAME\_RIS bit in DCMI\_RIS

### 23.7.7 DCMI embedded synchronization code register (DCMI\_ESCR)

Address offset: 0x18

Reset value: 0x0000 0000

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

Bits 31:24 **FEC[7:0]**: Frame end delimiter code

This byte specifies the code of the frame end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FEC.

If FEC is programmed to 0xFF, all the unused codes (0xFF0000XY) are interpreted as frame end delimiters.

Bits 23:16 **LEC[7:0]**: Line end delimiter code

This byte specifies the code of the line end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LEC.

Bits 15:8 **LSC[7:0]**: Line start delimiter code

This byte specifies the code of the line start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LSC.

Bits 7:0 **FSC[7:0]**: Frame start delimiter code

This byte specifies the code of the frame start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FSC.

If FSC is programmed to 0xFF, no frame start delimiter is detected. But, the 1<sup>st</sup> occurrence of LSC after an FEC code will be interpreted as a start of frame delimiter.

### 23.7.8 DCMI embedded synchronization unmask register (DCMI\_ESUR)

Address offset: 0x1C

Reset value: 0x0000 0000

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

Bits 31:24 **FEU[7:0]**: Frame end delimiter unmask  
 This byte specifies the mask to be applied to the code of the frame end delimiter.  
 0: The corresponding bit in the FEC byte in DCMI\_ESCR is masked while comparing the frame end delimiter with the received data.  
 1: The corresponding bit in the FEC byte in DCMI\_ESCR is compared while comparing the frame end delimiter with the received data

Bits 23:16 **LEU[7:0]**: Line end delimiter unmask  
 This byte specifies the mask to be applied to the code of the line end delimiter.  
 0: The corresponding bit in the LEC byte in DCMI\_ESCR is masked while comparing the line end delimiter with the received data  
 1: The corresponding bit in the LEC byte in DCMI\_ESCR is compared while comparing the line end delimiter with the received data

Bits 15:8 **LSU[7:0]**: Line start delimiter unmask  
 This byte specifies the mask to be applied to the code of the line start delimiter.  
 0: The corresponding bit in the LSC byte in DCMI\_ESCR is masked while comparing the line start delimiter with the received data  
 1: The corresponding bit in the LSC byte in DCMI\_ESCR is compared while comparing the line start delimiter with the received data

Bits 7:0 **FSU[7:0]**: Frame start delimiter unmask  
 This byte specifies the mask to be applied to the code of the frame start delimiter.  
 0: The corresponding bit in the FSC byte in DCMI\_ESCR is masked while comparing the frame start delimiter with the received data  
 1: The corresponding bit in the FSC byte in DCMI\_ESCR is compared while comparing the frame start delimiter with the received data

### 23.7.9 DCMI crop window start (DCMI\_CWSTRT)

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.	VST[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	HOFFCNT[13:0]													
		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 28:16 **VST[12:0]**: Vertical start line count

The image capture starts with this line number. Previous line data are ignored.

0x0000 => line 1

0x0001 => line 2

0x0002 => line 3

....

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **HOFFCNT[13:0]**: Horizontal offset count

This value gives the number of pixel clocks to count before starting a capture.

### 23.7.10 DCMI crop window size (DCMI\_CWSIZE)

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.	VLINE[13:0]													
		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.	Res.	CAPCNT[13:0]													
		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 29:16 **VLINE[13:0]**: Vertical line count

This value gives the number of lines to be captured from the starting point.

0x0000 => 1 line

0x0001 => 2 lines

0x0002 => 3 lines

....

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **CAPCNT[13:0]**: Capture count

This value gives the number of pixel clocks to be captured from the starting point on the same line. It value should corresponds to word-aligned data for different widths of parallel interfaces.

0x0000 => 1 pixel

0x0001 => 2 pixels

0x0002 => 3 pixels

....

### 23.7.11 DCMI data register (DCMI\_DR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Byte3[7:0]								Byte2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte1[7:0]								Byte0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **Byte3[7:0]**: Data byte 3

Bits 23:16 **Byte2[7:0]**: Data byte 2

Bits 15:8 **Byte1[7:0]**: Data byte 1

Bits 7:0 **Byte0[7:0]**: Data byte 0

The digital camera Interface packages all the received data in 32-bit format before requesting a DMA transfer. A 4-word deep FIFO is available to leave enough time for DMA transfers and avoid DMA overrun conditions.

23.7.12 DCMI register map

Table 147 summarizes the DCMI registers.

Table 147. DCMI 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	DCMI_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
	Reset value												0	0	0	0	0			ENABLE			EDM	0	0	0	0	VSPOL	HSPOL	PKPOL	ESS	JPEG	CROP	CM	CAPTURE		
0x04	DCMI_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																															FNE	VSYNC	HSYNC			
0x08	DCMI_RIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																															LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS	
0x0C	DCMI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value																															LINE_IE	VSYNC_IE	ERR_IE	OVR_IE	FRAME_IE	
0x10	DCMI_MIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																															LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS	
0x14	DCMI_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																															LINE_ISC	VSYNC_ISC	ERR_ISC	OVR_ISC	FRAME_ISC	
0x18	DCMI_ESCR	FEC								LEC								LSC								FSC											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x1C	DCMI_ESUR	FEU								LEU								LSU								FSU											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	DCMI_CWSTRT	Res.	Res.	Res.	VST[12:0]												Res.	Res.	HOFFCNT[13:0]																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x24	DCMI_CWSIZE	Res.	Res.	VLINE[13:0]												Res.	Res.	CAPCNT[13:0]																			
	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	
0x28	DCMI_DR	Byte3								Byte2								Byte1								Byte0											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	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 2.2.2 on page 82 for the register boundary addresses.



## 24 Voltage reference buffer (VREFBUF)

### 24.1 Introduction

The STM32L4Rxxx and STM32L4Sxxx devices embed a voltage reference buffer which can be used as voltage reference for ADCs, DACs 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).

### 24.2 VREFBUF functional description

The internal voltage reference buffer supports two voltages<sup>(a)</sup>, which are configured with VRS bits in the VREFBUF\_CSR register:

- VRS = 0: V<sub>REF\_OUT1</sub> around 2.048 V.
- VRS = 1: V<sub>REF\_OUT2</sub> around 2.5 V.

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:

**Table 148. VREF buffer modes**

ENVR	HIZ	VREF buffer configuration
0	0	VREFBUF buffer OFF: – 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: – VREFBUF buffer OFF – V <sub>REF+</sub> pin floating. The voltage is held with the external capacitor – VRR detection disabled and VRR bit keeps last state

After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF\_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

a. The minimum V<sub>DDA</sub> voltage depends on VRS setting, refer to the product datasheet.

## 24.3 VREFBUF registers

### 24.3.1 VREFBUF control and status register (VREFBUF\_CSR)

Address offset: 0x00

Reset value: 0x0000 0002

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

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

Bit 3 **VRR**: Voltage reference buffer ready

0: the voltage reference buffer output is not ready.

1: the voltage reference buffer output reached the requested level.

Bit 2 **VRS**: Voltage reference scale

This bit selects the value generated by the voltage reference buffer.

0: Voltage reference set to  $V_{REF\_OUT1}$  (around 2.048 V).

1: Voltage reference set to  $V_{REF\_OUT2}$  (around 2.5 V).

Bit 1 **HIZ**: High impedance mode

This bit controls the analog switch to connect or not the  $V_{REF+}$  pin.

0:  $V_{REF+}$  pin is internally connected to the voltage reference buffer output.

1:  $V_{REF+}$  pin is high impedance.

Refer to [Table 148: 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.



### 24.3.2 VREFBUF calibration control register (VREFBUF\_CCR)

Address offset: 0x04

Reset value: 0x0000 00XX

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

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

Bits 5:0 **TRIM[5:0]**: Trimming code

These bits are automatically initialized after reset with the trimming value stored in the Flash memory during the production test. Writing into these bits allows to tune the internal reference buffer voltage.

### 24.3.3 VREFBUF register map

The following table gives the VREFBUF register map and the reset values.

**Table 149. VREFBUF register map and reset values**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
0x00	VREFBUF_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRR	VRS	HIZ	ENVR					
	Reset value																													0	0	1	0					
0x04	VREFBUF_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]									
	Reset value																												x	x	x	x	x	x				

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.

## 25 Comparator (COMP)

### 25.1 Introduction

The device embeds 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.

### 25.2 COMP main features

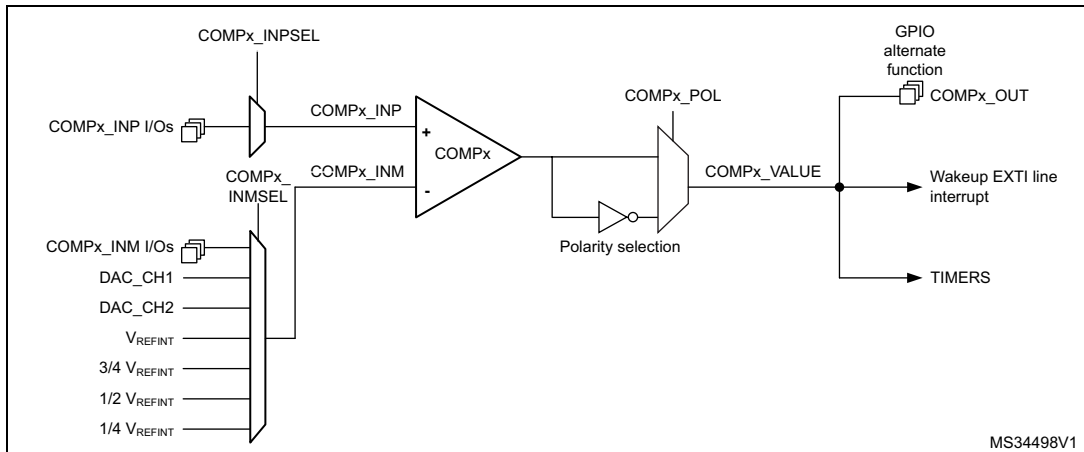
- Each comparator has configurable plus and minus inputs used for flexible voltage selection:
  - Multiplexed I/O pins
  - DAC Channel1 and Channel2
  - Internal reference voltage and three submultiple values (1/4, 1/2, 3/4) provided by scaler (buffered voltage divider)
- Programmable hysteresis
- Programmable speed / consumption
- The outputs can be redirected to an I/O or to timer inputs for triggering:
  - Break events for fast PWM shutdowns
- Comparator outputs with blanking source
- The two comparators can be combined in a window comparator
- Each comparator has interrupt generation capability with wake-up from Sleep and Stop modes (through the EXTI controller)

## 25.3 COMP functional description

### 25.3.1 COMP block diagram

The block diagram of the comparators is shown in [Figure 155: Comparators block diagram](#).

**Figure 155. Comparators block diagram**



### 25.3.2 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 150. COMP1 input plus assignment**

COMP1_INP	COMP1_INPSEL
PC5	0
PB2	1

**Table 151. COMP1 input minus assignment**

COMP1_INM	COMP1_INMSEL[2:0]
$\frac{1}{4} V_{REFINT}$	000
$\frac{1}{2} V_{REFINT}$	001

**Table 151. COMP1 input minus assignment (continued)**

COMP1_INM	COMP1_INMSEL[2:0]
$\frac{3}{4} V_{REFINT}$	010
$V_{REFINT}$	011
DAC Channel1	100
DAC Channel2	101
PB1	110
PC4	111

**Table 152. COMP2 input plus assignment**

COMP2_INP	COMP2_INPSEL
PB4	0
PB6	1

**Table 153. COMP2 input minus assignment**

COMP2_INM	COMP2_INMSEL[2:0]
$\frac{1}{4} V_{REFINT}$	000
$\frac{1}{2} V_{REFINT}$	001
$\frac{3}{4} V_{REFINT}$	010
$V_{REFINT}$	011
DAC Channel1	100
DAC Channel2	101
PB3	110
PB7	111

### 25.3.3 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.

*Note:* **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.

### 25.3.4 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.

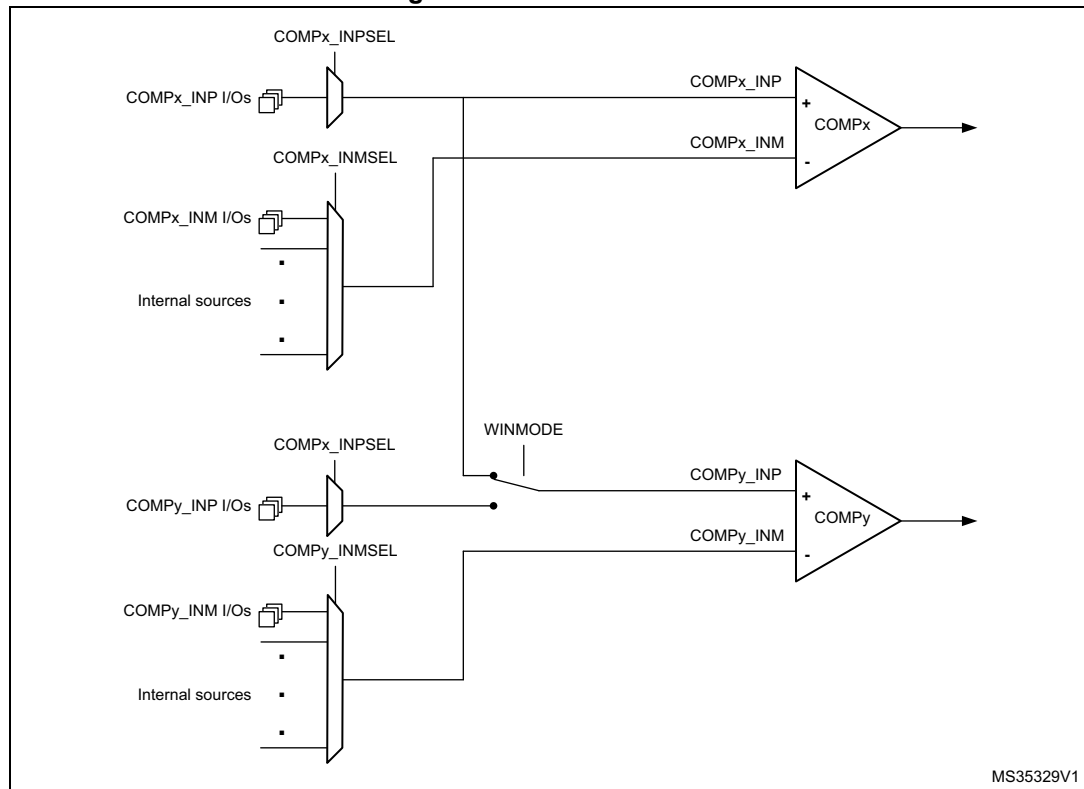
The write protection can only be reset by a MCU reset.

### 25.3.5 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 156. Window mode

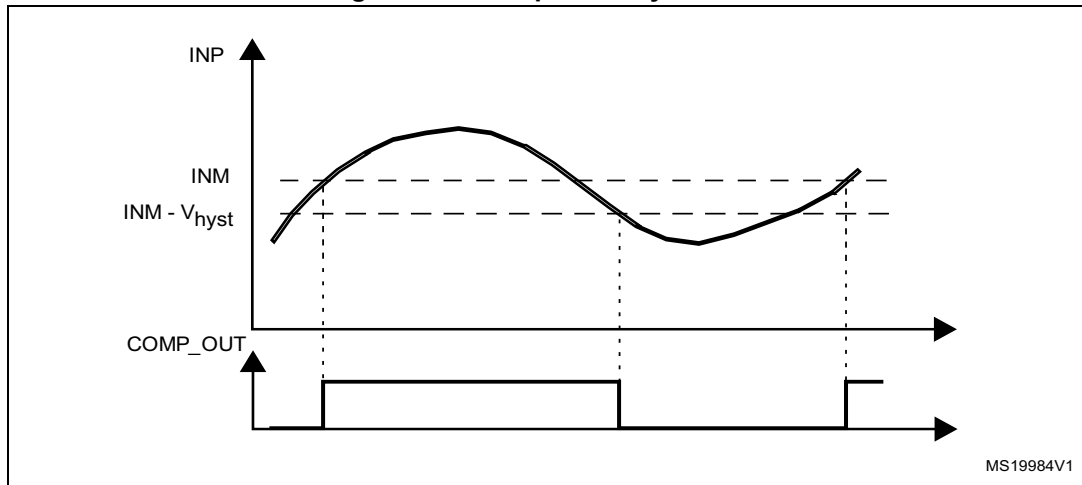


MS35329V1

### 25.3.6 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.

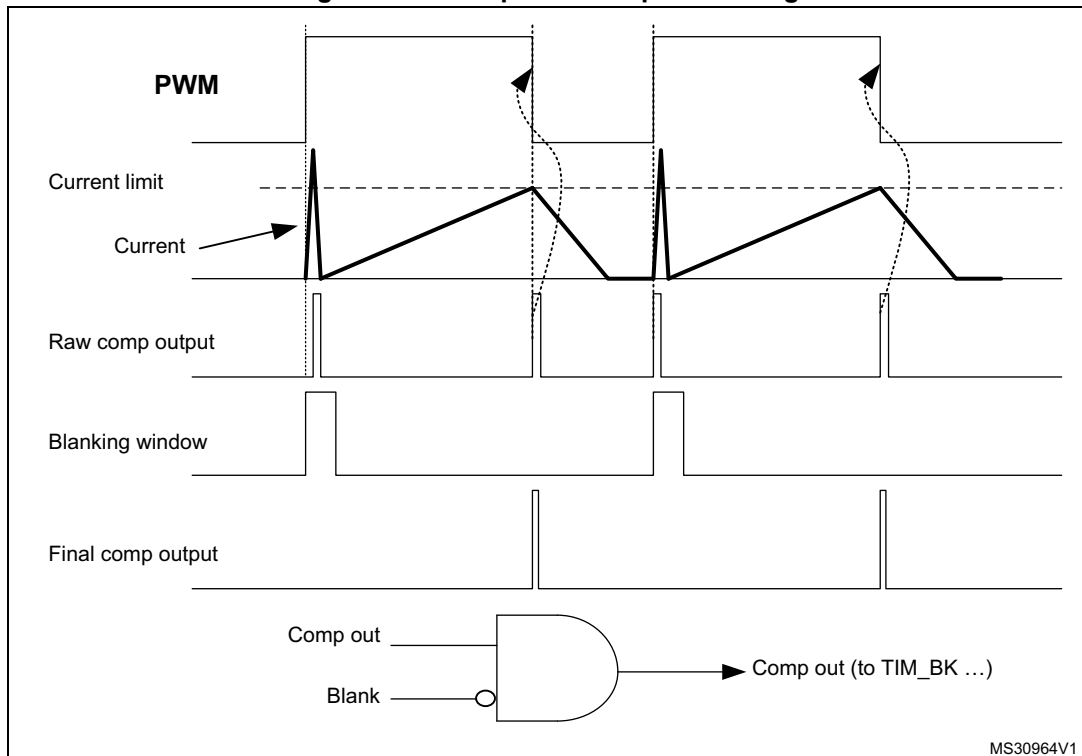
Figure 157. Comparator hysteresis



### 25.3.7 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 158. Comparator output blanking



### 25.3.8 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

## 25.4 COMP low-power modes

**Table 154. 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	

## 25.5 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 the 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 the COMPx

Table 155. 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

## 25.6 COMP registers

### 25.6.1 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.	Res.	Res.	Res.	SCAL EN	BRG EN	Res.	BLANKING		HYST		
rs	r							rw	rw		rw		rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLA RITY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INP SEL.	INMSEL		PWRMODE		Res.	EN	
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: 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: Reserved, must be kept at reset value.
- Bit 7 **INPSEL**: Comparator1 input plus selection bit  
This bit is set and cleared by software (only if LOCK not set).  
0: external IO - PC5  
1: PB2
- Bits 6:4 **INMSEL**: 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 = DAC Channel1  
101 = DAC Channel2  
110 = PB1111 = PC4

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 cleared.

Bit 0 **EN**: Comparator 1 enable bit  
 This bit is set and cleared by software (only if LOCK not set). It switches on Comparator 1.  
 0: Comparator 1 switched OFF  
 1: Comparator 1 switched ON

### 25.6.2 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.	Res.	Res.	Res.	SCAL EN	BRG EN	Res.	BLANKING		HYST		
rs	r							rw	rw		rw		rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLA RITY	Res.	Res.	Res.	Res.	Res.	WIN MODE	Res.	INP SEL.	INMSEL			PWRMODE		Res.	EN
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: 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<sub>REFINT</sub> 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  
100: TIM15 OC1 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 will be 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
- Bit 8 Reserved, must be kept at reset value.
- Bit 7 **INPSEL**: Comparator 1 input plus selection bit  
This bit is set and cleared by software (only if LOCK not set).  
0: PB4  
1: PB6

Bits 6:4 **INMSEL**: 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 = DAC Channel1

101 = DAC Channel2

110 = PB3

111 = PB7

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 cleared.

Bit 0 **EN**: Comparator 2 enable bit

This bit is set and cleared by software (only if LOCK not set). It switches on comparator2.

0: Comparator 2 switched OFF

1: Comparator 2 switched ON

### 25.6.3 COMP register map

The following table summarizes the comparator registers.

**Table 156. COMP register map and reset values**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	COMP1_CSR	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	Res.	SCALEN	BRGEN	Res.	BLANKING			HYST		POLARITY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INMSEL			PWRRMODE		Res.	EN	
	Reset value	0	0							0	0		0	0	0	0	0	0	0							0	0	0	0	0	0		0	
0x04	COMP2_CSR	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	Res.	SCALEN	BRGEN	Res.	BLANKING			HYST		POLARITY	Res.	Res.	Res.	Res.	Res.	WINMODE	Res.	Res.	INPSEL	INMSEL			PWRRMODE		Res.	EN
	Reset value	0	0							0	0		0	0	0	0	0	0	0					0		0	0	0	0	0	0		0	

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.

## 26 Operational amplifiers (OPAMP)

### 26.1 Introduction

The device embeds two operational amplifiers with two inputs and one output each. The three I/Os can be connected to the external pins, this enables any type of external interconnections. The operational amplifier can be configured internally as a follower or as an amplifier with a non-inverting gain ranging from 2 to 16.

The positive input can be connected to the internal DAC.

The output can be connected to the internal ADC.

### 26.2 OPAMP main features

- Rail-to-rail input and output voltage range
- Low input bias current (down to 1 nA)
- Low input offset voltage (1.5 mV after calibration, 3 mV with factory calibration)
- Low-power mode (current consumption reduced to 30  $\mu$ A instead of 100  $\mu$ A)
- Fast wakeup time (10  $\mu$ s in normal mode, 30  $\mu$ s in low-power mode)
- Gain bandwidth of 1.6 MHz

### 26.3 OPAMP functional description

The OPAMP has several modes.

Each OPAMP can be individually enabled, when disabled the output is high-impedance.

When enabled, it can be in calibration mode, all input and output of the OPAMP are then disconnected, or in functional mode.

There are two functional modes, the low-power mode or the normal mode. In functional mode the inputs and output of the OPAMP are connected as described in the [Section 26.3.3: Signal routing](#).

#### 26.3.1 OPAMP reset and clocks

The operational amplifier clock is necessary for accessing the registers. When the application does not need to have read or write access to those registers, the clock can be switched off using the peripheral clock enable register (see OPAMPEN bit in [Section 6.4.19: APB1 peripheral clock enable register 1 \(RCC\\_APB1ENR1\)](#)).

The bit OPAEN enables and disables the OPAMP operation. The OPAMP registers configurations should be changed before enabling the OPAEN bit in order to avoid spurious effects on the output.

When the output of the operational amplifier is no more needed the operational amplifier can be disabled to save power. All the configurations previously set (including the calibration) are maintained while OPAMP is disabled.

### 26.3.2 Initial configuration

The default configuration of the operational amplifier is a functional mode where the three IOs are connected to external pins. In the default mode the operational amplifier uses the factory trimming values. See electrical characteristics section of the datasheet for factory trimming conditions, usually the temperature is 30 °C and the voltage is 3 V. The trimming values can be adjusted, see [Section 26.3.5: Calibration](#) for changing the trimming values. The default configuration uses the normal mode, which provides the highest performance. Bit OPALPM can be set in order to switch the operational amplifier to low-power mode and reduced performance. Both normal and low-power mode characteristics are defined in the section “electrical characteristics” of the datasheet. Before utilization, the bit OPA\_RANGE of OPAMP\_CSR must be set to 1 if  $V_{DDA}$  is above 2.4V, or kept at 0 otherwise.

As soon as the OPAEN bit in OPAMP\_CSR register is set, the operational amplifier is functional. The two input pins and the output pin are connected as defined in [Section 26.3.3: Signal routing](#) and the default connection settings can be changed.

*Note:* The inputs and output pins must be configured in analog mode (default state) in the corresponding GPIOx\_MODER register.

### 26.3.3 Signal routing

The routing for the operational amplifier pins is determined by OPAMP\_CSR register.

The connections of the two operational amplifiers (OPAMP1 and OPAMP2) are described in the table below

**Table 157. Operational amplifier possible connections**

Signal	Pin	Internal	comment
OPAMP1_VINM	PA1 or dedicated pin <sup>(1)</sup>	OPAMP1_OUT or PGA	Controlled by bits OPAMODE and VM_SEL.
OPAMP1_VINP	PA0	DAC1_OUT1	Controlled by bit VP_SEL.
OPAMP1_VOUT	PA3	ADC1_IN8	The pin is connected when the OPAMP is enabled. The ADC input is controlled by ADC.
OPAMP2_VINM	PA7 or dedicated pin <a href="#">Chapter 1</a> .	OPAMP2_OUT or PGA	Controlled by bits OPAMODE and VM_SEL.
OPAMP2_VINP	PA6	DAC1_OUT2	Controlled by bit VP_SEL
OPAMP2_VOUT	PB0	ADC1_IN15	The pin is connected when the OPAMP is enabled. The ADC input is controlled by ADC.

1. The dedicated pin is only available on BGA132 and BGA169 package. This configuration provides the lowest input bias current (see datasheet).

### 26.3.4 OPAMP modes

The operational amplifier inputs and outputs are all accessible on terminals. The amplifiers can be used in multiple configuration environments:

- Standalone mode (external gain setting mode)
- Follower configuration mode
- PGA modes

*Note:* The amplifier output pin is directly connected to the output pad to minimize the output impedance. It cannot be used as a general purpose I/O, even if the amplifier is configured as a PGA and only connected to the ADC channel.

*Note:* The impedance of the signal must be maintained below a level which avoids the input leakage to create significant artifacts (due to a resistive drop in the source). Please refer to the electrical characteristics section in the datasheet for further details.

#### Standalone mode (external gain setting mode)

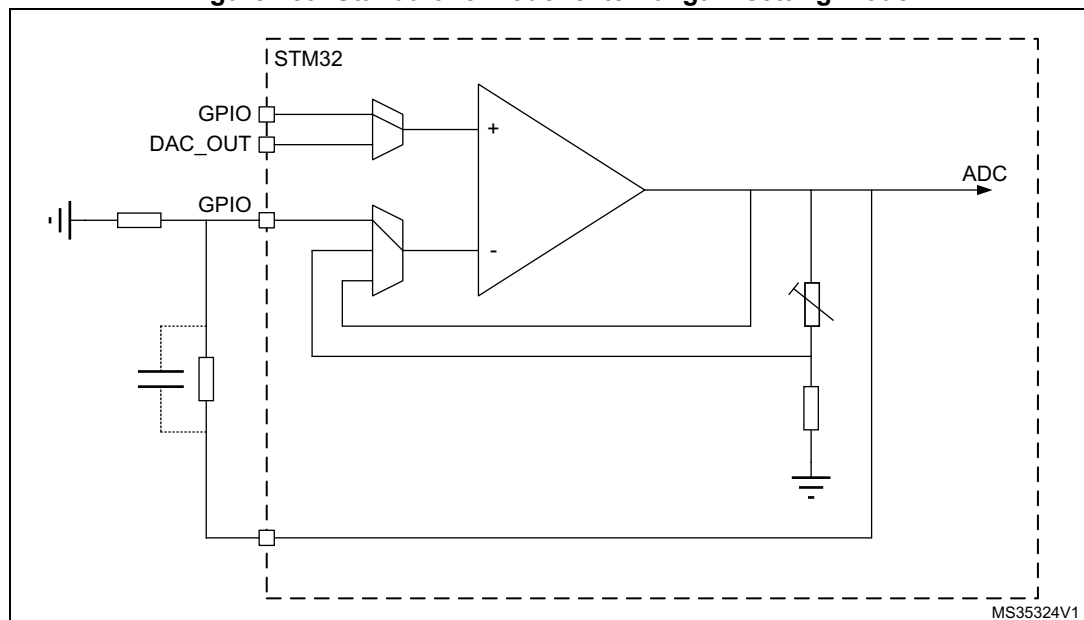
The procedure to use the OPAMP in standalone mode is presented hereafter.

Starting from the default value of OPAMP\_CSR, and the default state of GPIOx\_MODER, configure bit OPA\_RANGE according the  $V_{DDA}$  voltage. As soon as the OPAEN bit is set, the two input pins and the output pin are connected to the operational amplifier.

This default configuration uses the factory trimming values and operates in normal mode (highest performance). The behavior of the OPAMP can be changed as follows:

- OPALPM can be set to “operational amplifier low-power” mode in order to save power.
- USERTRIM can be set to modify the trimming values for the input offset.

**Figure 159. Standalone mode: external gain setting mode**





**Follower configuration mode**

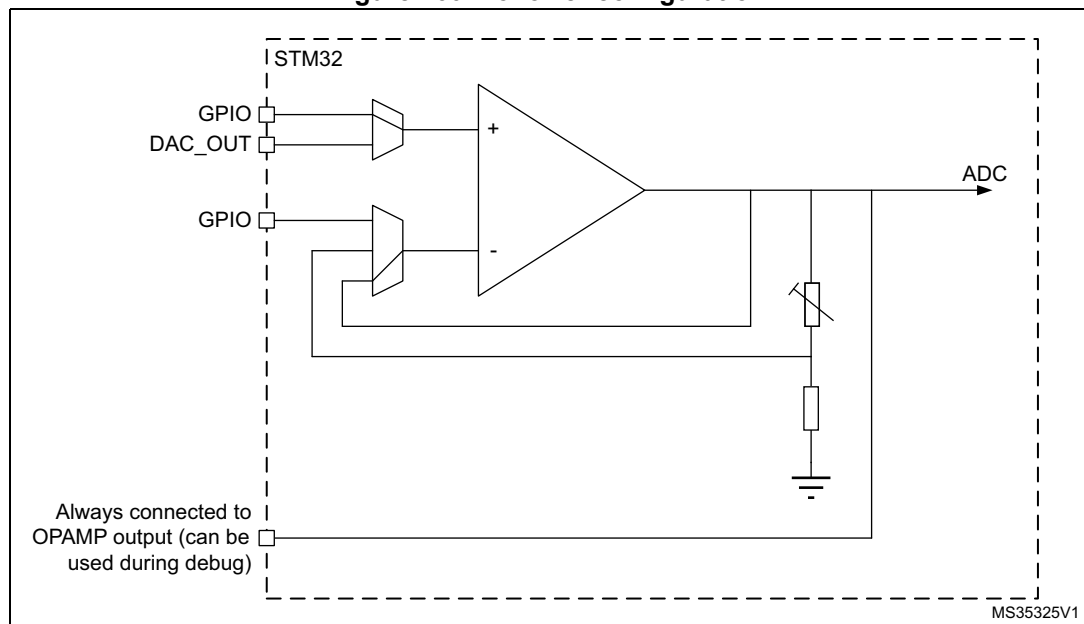
The procedure to use the OPAMP in follower mode is presented hereafter.

- configure OPAMODE bits as “internal follower”
- configure VP\_SEL bits as “GPIO connected to VINP”.
- As soon as the OPAEN bit is set, the signal on pin OPAMP\_VINP is copied to pin OPAMP\_VOUT.

*Note:* The pin corresponding to OPAMP\_VINM is free for another usage.

*Note:* The signal on the operational amplifier output is also seen as an ADC input. As a consequence, the OPAMP configured in follower mode can be used to perform impedance adaptation on input signals before feeding them to the ADC input, assuming the input signal frequency is compatible with the operational amplifier gain bandwidth specification.

**Figure 160. Follower configuration**



### Programmable Gain Amplifier mode

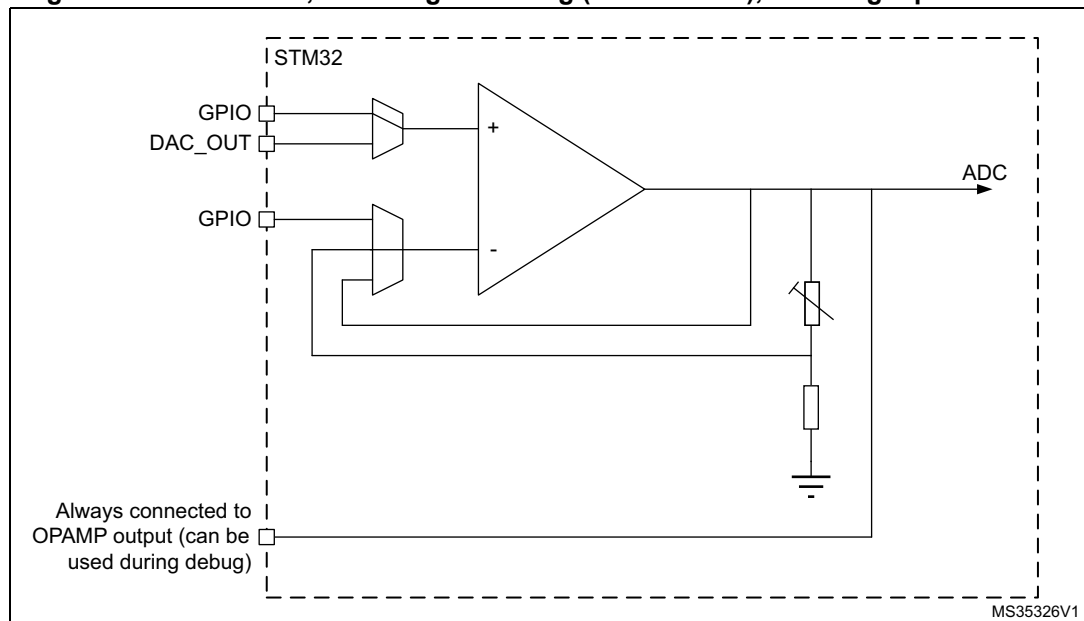
The procedure to use the OPAMP to amplify the amplitude of an input signal is presented hereafter.

- configure OPAMODE bits as “internal PGA enabled”,
- configure PGA\_GAIN bits as “internal PGA Gain 2, 4, 8 or 16”,
- configure VM\_SEL bits as “inverting not externally connected”,
- configure VP\_SEL bits as “GPIO connected to VINP”.

As soon as the OPAEN bit is set, the signal on pin OPAMP\_VINP is amplified by the selected gain and visible on pin OPAMP\_VOUT.

*Note:* To avoid saturation, the input voltage should stay below  $V_{DDA}$  divided by the selected gain.

**Figure 161. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used**



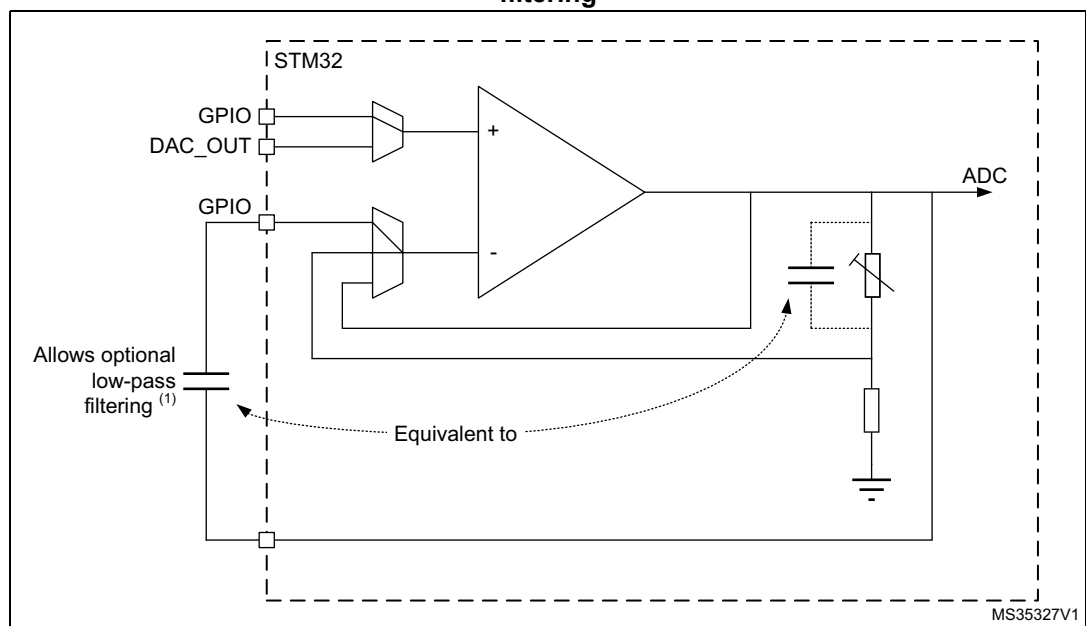
**Programmable Gain Amplifier mode with external filtering**

The procedure to use the OPAMP to amplify the amplitude of an input signal, with an external filtering, is presented hereafter.

- configure OPAMODE bits as “internal PGA enabled”,
- configure PGA\_GAIN bits as “internal PGA Gain 2, 4, 8 or 16”,
- configure VM\_SEL bits as “GPIO connected to VINM”,
- configure VP\_SEL bits as “GPIO connected to VINP”.

Any external connection on VINP can be used in parallel with the internal PGA, for example a capacitor can be connected between VOUT and VINM for filtering purpose (see datasheet for the value of resistors used in the PGA resistor network).

**Figure 162. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering**



1. The gain depends on the cut-off frequency.

**26.3.5 Calibration**

At startup, the trimming values are initialized with the preset ‘factory’ trimming value.

Each operational amplifier offset can be trimmed by the user. Specific registers allow to have different trimming values for normal mode and for low-power mode.

The aim of the calibration is to cancel as much as possible the OPAMP inputs offset voltage. The calibration circuitry allows to reduce the inputs offset voltage to less than +/-1.5 mV within stable voltage and temperature conditions.

For each operational amplifier and each mode two trimming values need to be trimmed, one for N differential pair and one for P differential pair.

There are two registers for trimming the offsets for each operational amplifiers, one for normal mode (OPAMP\_OTR) and one low-power mode (OPAMP\_LPOTR). Each register is composed of five bits for P differential pair trimming and five bits for N differential pair trimming. These are the ‘user’ values.

The user is able to switch from 'factory' values to 'user' trimmed values using the USERTRIM bit in the OPAMP\_CSR register. This bit is reset at startup and so the 'factory' value are applied by default to the OPAMP trimming registers.

User is liable to change the trimming values in calibration or in functional mode.

The offset trimming registers are typically configured after the calibration operation is initialized by setting bit CALON to 1. When CALON = 1 the inputs of the operational amplifier are disconnected from the functional environment.

- Setting CALSEL to 1 initializes the offset calibration for the P differential pair (low voltage reference used).
- Resetting CALSEL to 0 initializes the offset calibration for the N differential pair (high voltage reference used).

When CALON = 1, the bit CALOUT will reflect the influence of the trimming value selected by CALSEL and OPALPM. When the value of CALOUT switches between two consecutive trimming values, this means that those two values are the best trimming values. The CALOUT flag needs up to 1 ms after the trimming value is changed to become steady (see  $t_{OFFTRIMmax}$  delay specification in the electrical characteristics section of the datasheet).

*Note: The closer the trimming value is to the optimum trimming value, the longer it takes to stabilize (with a maximum stabilization time remaining below 1 ms in any case).*

**Table 158. Operating modes and calibration**

Mode	Control bits				Output	
	OPAEN	OPALPM	CALON	CALSEL	V <sub>OUT</sub>	CALOUT flag
Normal operating mode	1	0	0	X	analog	0
Low-power mode	1	1	0	X	analog	0
Power down	0	X	X	X	Z	0
Offset cal high for normal mode	1	0	1	0	analog	X
Offset cal low for normal mode	1	0	1	1	analog	X
Offset cal high for low-power mode	1	1	1	0	analog	X
Offset cal low for low-power mode	1	1	1	1	analog	X

### Calibration procedure

Here are the steps to perform a full calibration of either one of the operational amplifiers:

1. Select correct OPA\_RANGE in OPAMP\_CSR, then set the OPAEN bit in OPAMP\_CSR to 1 to enable the operational amplifier.
2. Set the USERTRIM bit in the OPAMP\_CSR register to 1.
3. Choose a calibration mode (refer to [Table 158: Operating modes and calibration](#)). The steps 3 to 4 will have to be repeated 4 times. For the first iteration select
  - Normal mode, offset cal high (N differential pair)
 The above calibration mode correspond to OPALPM=0 and CALSEL=0 in the OPAMP\_CSR register.
4. Increment TRIMOFFSETN[4:0] in OPAMP\_OTR starting from 00000b until CALOUT changes to 1 in OPAMP\_CSR.

*Note:* CALOUT will switch from 0 to 1 for offset cal high and from 1 to 0 for offset cal low.

*Note:* Between the write to the OPAMP\_OTR register and the read of the CALOUT value, make sure to wait for the  $t_{OFFTRIM,max}$  delay specified in the electrical characteristics section of the datasheet, to get the correct CALOUT value.

The commutation means that the offset is correctly compensated and that the corresponding trim code must be saved in the OPAMP\_OTR register.

Repeat steps 3 to 4 for:

- Normal\_mode and offset cal low
- Low power mode and offset cal high
- Low power mode and offset cal low

If a mode is not used it is not necessary to perform the corresponding calibration.

All operational amplifier can be calibrated at the same time.

*Note:* During the whole calibration phase the external connection of the operational amplifier output must not pull up or down currents higher than 500  $\mu$ A.

During the calibration procedure, it is necessary to set up OPAMODE bits as 00 or 01 (PGA disable) or 11 (internal follower).

## 26.4 OPAMP low-power modes

**Table 159. Effect of low-power modes on the OPAMP**

Mode	Description
Sleep	No effect.
Low-power run	No effect.
Low-power sleep	No effect.
Stop 0 / Stop 1	No effect, OPAMP registers content is kept.
Stop 2	OPAMP registers content is kept. OPAMP must be disabled before entering Stop 2 mode.

Table 159. Effect of low-power modes on the OPAMP (continued)

Mode	Description
Standby	The OPAMP registers are powered down and must be re-initialized after exiting Standby or Shutdown mode.
Shutdown	

## 26.5 OPAMP registers

### 26.5.1 OPAMP1 control/status register (OPAMP1\_CSR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPA_RANGE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r/w															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAL_OUT	USER_TRIM	CAL_SEL	CALON	Res.	VP_SEL	VM_SEL		Res.	Res.	PGA_GAIN		OPAMODE		OPA_LPM	OPAEN
r	r/w	r/w	r/w		r/w	r/w	r/w			r/w	r/w	r/w	w	r/w	r/w

Bit 31 **OPA\_RANGE**: Operational amplifier power supply range for stability

All AOP must be in power down to allow AOP-RANGE bit write. It applies to all AOP embedded in the product.

0: Low range (VDDA < 2.4V)

1: High range (VDDA > 2.4V)

Bits 30:16 Reserved, must be kept at reset value.

Bit 15 **CALOUT**: Operational amplifier calibration output

During calibration mode offset is trimmed when this signal toggle.

Bit 14 **USERTRIM**: allows to switch from 'factory' AOP offset trimmed values to AOP offset 'user' trimmed values

This bit is active for both mode normal and low-power.

0: 'factory' trim code used

1: 'user' trim code used

Bit 13 **CALSEL**: Calibration selection

0: NMOS calibration (200mV applied on OPAMP inputs)

1: PMOS calibration (VDDA-200mV applied on OPAMP inputs)

Bit 12 **CALON**: Calibration mode enabled

0: Normal mode

1: Calibration mode (all switches opened by HW)

Bit 11 Reserved, must be kept at reset value.

Bit 10 **VP\_SEL**: Non inverted input selection

0: GPIO connected to VINP

1: DAC connected to VINP

- Bits 9:8 **VM\_SEL**: Inverting input selection  
 These bits are used only when OPAMODE = 00, 01 or 10.  
 00: GPIO connected to VINM (valid also in PGA mode for filtering)  
 01: Dedicated low leakage input, connected to VINM (valid also in PGA mode for filtering)  
 1x: Inverting input not externally connected. These configurations are valid only when OPAMODE = 10 (PGA mode)
- Bits 7:6 Reserved, must be kept at reset value.
- Bits 5:4 **PGA\_GAIN**: Operational amplifier Programmable amplifier gain value  
 00: internal PGA Gain 2  
 01: internal PGA Gain 4  
 10: internal PGA Gain 8  
 11: internal PGA Gain 16
- Bits 3:2 **OPAMODE**: Operational amplifier PGA mode  
 00: internal PGA disable  
 01: internal PGA disable  
 10: internal PGA enable, gain programmed in PGA\_GAIN  
 11: internal follower
- Bit 1 **OPALPM**: Operational amplifier Low Power Mode  
 The operational amplifier must be disable to change this configuration.  
 0: operational amplifier in normal mode  
 1: operational amplifier in low-power mode
- Bit 0 **OPAEN**: Operational amplifier Enable  
 0: operational amplifier disabled  
 1: operational amplifier enabled

### 26.5.2 OPAMP1 offset trimming register in normal mode (OPAMP1\_OTR)

Address offset: 0x04

Reset value: 0x0000 XXXX (factory trimmed values)

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

- Bits 31:13 Reserved, must be kept at reset value.
- Bits 12:8 **TRIMOFFSETP[4:0]**: Trim for PMOS differential pairs
- Bits 7:5 Reserved, must be kept at reset value.
- Bits 4:0 **TRIMOFFSETN[4:0]**: Trim for NMOS differential pairs

### 26.5.3 OPAMP1 offset trimming register in low-power mode (OPAMP1\_LPOTR)

Address offset: 0x08



Reset value: 0x0000 XXXX (factory trimmed values)

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

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

Bits 12:8 **TRIMLPOFFSETP[4:0]**: Low-power mode trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMLPOFFSETN[4:0]**: Low-power mode trim for NMOS differential pairs

### 26.5.4 OPAMP2 control/status register (OPAMP2\_CRS)

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
CALOUT	USERTRIM	CALSEL	CALON	Res.	VP_SEL	VM_SEL		Res.	Res.	PGA_GAIN		OPAMODE		OPALPM	OPAEN
r	rw	rw	rw		rw	rw	rw			rw	rw	rw	rw	rw	rw

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

Bit 15 **CALOUT**: Operational amplifier calibration output  
During calibration mode offset is trimmed when this signal toggle.

Bit 14 **USERTRIM**: allows to switch from 'factory' AOP offset trimmed values to AOP offset 'user' trimmed values  
This bit is active for both mode normal and low-power.  
0: 'factory' trim code used  
1: 'user' trim code used

Bit 13 **CALSEL**: Calibration selection  
0: NMOS calibration (200mV applied on OPAMP inputs)  
1: PMOS calibration (VDDA-200mV applied on OPAMP inputs)

Bit 12 **CALON**: Calibration mode enabled  
0: Normal mode  
1: Calibration mode (all switches opened by HW)

Bit 11 Reserved, must be kept at reset value.

Bit 10 **VP\_SEL**: Non inverted input selection  
0: GPIO connected to VINP  
1: DAC connected to VINP



- Bits 9:8 **VM\_SEL**: Inverting input selection  
 These bits are used only when OPAMODE = 00, 01 or 10.  
 00:GPIO connected to VINM (valid also in PGA mode for filtering)  
 01: Dedicated low leakage input, (available only on BGA132) connected to VINM (valid also in PGA mode for filtering)  
 1x: Inverting input not externally connected. These configurations are valid only when OPAMODE = 10 (PGA mode)
- Bits 7:6 Reserved, must be kept at reset value.
- Bits 5:4 **PGA\_GAIN**: Operational amplifier Programmable amplifier gain value  
 00: internal PGA Gain 2  
 01: internal PGA Gain 4  
 10: internal PGA Gain 8  
 11: internal PGA Gain 16
- Bits 3:2 **OPAMODE**: Operational amplifier PGA mode  
 00: internal PGA disable  
 01: internal PGA disable  
 10: internal PGA enable, gain programmed in PGA\_GAIN  
 11: internal follower
- Bit 1 **OPALPM**: Operational amplifier Low Power Mode  
 The operational amplifier must be disable to change this configuration.  
 0: operational amplifier in normal mode  
 1: operational amplifier in low-power mode
- Bit 0 **OPAEN**: Operational amplifier Enable  
 0: operational amplifier disabled  
 1: operational amplifier enabled

### 26.5.5 OPAMP2 offset trimming register in normal mode (OPAMP2\_OTR)

Address offset: 0x14

Reset value: 0x0000 XXXX (factory trimmed values)

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

- Bits 31:13 Reserved, must be kept at reset value.
- Bits 12:8 **TRIMOFFSETP[4:0]**: Trim for PMOS differential pairs
- Bits 7:5 Reserved, must be kept at reset value.
- Bits 4:0 **TRIMOFFSETN[4:0]**: Trim for NMOS differential pairs

### 26.5.6 OPAMP2 offset trimming register in low-power mode (OPAMP2\_LPOTR)

Address offset: 0x18



Reset value: 0x0000 XXXX (factory trimmed values)

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

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

Bits 12:8 **TRIMLPOFFSETP[4:0]**: Low-power mode trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMLPOFFSETN[4:0]**: Low-power mode trim for NMOS differential pairs

### 26.5.7 OPAMP register map

Table 160. OPAMP register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	OPAMP1_CSR	OPA_RANGE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALOUT	USERTRIM	CALSEL	CALON	Res.	VP_SEL	VM_SEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value	0																	0	0	0	0		0	0	0			0	0	0	0	0	0
0x04	OPAMP1_OTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM OFFSETP[4:0]				Res.	Res.	Res.	Res.	TRIM OFFSETN[4:0]				
	Reset value																						(1)								(1)			
0x08	OPAMP1_LPOTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIMLP OFFSETP[4:0]				Res.	Res.	Res.	Res.	TRIMLP OFFSETN[4:0]				
	Reset value																						(1)								(1)			
0x10	OPAMP2_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALOUT	USERTRIM	CALSEL	CALON	Res.	VP_SEL	VM_SEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																			0	0	0	0		0	0	0			0	0	0	0	0
0x14	OPAMP2_OTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM OFFSETP[4:0]				Res.	Res.	Res.	Res.	TRIM OFFSETN[4:0]			
	Reset value																							(1)								(1)		
0x18	OPAMP2_LPO TR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIMLP OFFSETP[4:0]				Res.	Res.	Res.	Res.	TRIMLP OFFSETN[4:0]			
	Reset value																							(1)								(1)		

1. Factory trimmed values.

Refer to [Section 2.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.



## 27 Digital filter for sigma delta modulators (DFSDM)

### 27.1 Introduction

Digital filter for sigma delta modulators (DFSDM) is a high-performance module dedicated to interface external  $\Sigma\Delta$  modulators to a microcontroller. It is featuring up to 8 external digital serial interfaces (channels) and up to 4 digital filters with flexible Sigma Delta stream digital processing options to offer up to 24-bit final ADC resolution. DFSDM also features optional parallel data stream input from internal ADC peripherals or from microcontroller memory.

An external  $\Sigma\Delta$  modulator provides digital data stream of converted analog values from the external  $\Sigma\Delta$  modulator analog input. This digital data stream is sent into a DFSDM input channel through a serial interface. DFSDM supports several standards to connect various  $\Sigma\Delta$  modulator outputs: SPI interface and Manchester coded 1-wire interface (both with adjustable parameters). DFSDM module supports the connection of up to 8 multiplexed input digital serial channels which are shared with up to 4 DFSDM modules. DFSDM module also supports alternative parallel data inputs from up to 8 internal 16-bit data channels (from internal ADCs or from microcontrollers memory).

DFSDM is converting an input data stream into a final digital data word which represents an analog input value on a  $\Sigma\Delta$  modulator analog input. The conversion is based on a configurable digital process: the digital filtering and decimation of the input serial data stream.

The conversion speed and resolution are adjustable according to configurable parameters for digital processing: filter type, filter order, length of filter, integrator length. The maximum output data resolution is up to 24 bits. There are two conversion modes: single conversion mode and continuous mode. The data can be automatically stored in a system RAM buffer through DMA, thus reducing the software overhead.

A flexible timer triggering system can be used to control the start of conversion of DFSDM. This timing control is capable of triggering simultaneous conversions or inserting a programmable delay between conversions.

DFSDM features an analog watchdog function. Analog watchdog can be assigned to any of the input channel data stream or to final output data. Analog watchdog has its own digital filtering of input data stream to reach the required speed and resolution of watched data.

To detect short-circuit in control applications, there is a short-circuit detector. This block watches each input channel data stream for occurrence of stable data for a defined time duration (several 0's or 1's in an input data stream).

An extremes detector block watches final output data and stores maximum and minimum values from the output data values. The extremes values stored can be restarted by software.

Two power modes are supported: normal mode and stop mode.

## 27.2 DFSDM main features

- Up to 8 multiplexed input digital serial channels:
  - configurable SPI interface to connect various  $\Sigma\Delta$  modulators
  - configurable Manchester coded 1 wire interface support
  - clock output for  $\Sigma\Delta$  modulator(s)
- Alternative inputs from up to 8 internal digital parallel channels:
  - inputs with up to 16 bit resolution
  - internal sources: ADCs data or memory (CPU/DMA write) data streams
- Adjustable digital signal processing:
  - Sinc<sup>X</sup> filter: filter order/type (1..5), oversampling ratio (up to 1..1024)
  - integrator: oversampling ratio (1..256)
- Up to 24-bit output data resolution:
  - right bit-shifter on final data (0..31 bits)
- Signed output data format
- Automatic data offset correction (offset stored in register by user)
- Continuous or single conversion
- Start-of-conversion synchronization with:
  - software trigger
  - internal timers
  - external events
  - start-of-conversion synchronously with first DFSDM filter (DFSDM\_FLT0)
- Analog watchdog feature:
  - low value and high value data threshold registers
  - own configurable Sinc<sup>X</sup> digital filter (order = 1..3, oversampling ratio = 1..32)
  - input from output data register or from one or more input digital serial channels
  - continuous monitoring independently from standard conversion
- Short-circuit detector to detect saturated analog input values (bottom and top ranges):
  - up to 8-bit counter to detect 1..256 consecutive 0's or 1's on input data stream
  - monitoring continuously each channel (8 serial channel transceiver outputs)
- Break generation on analog watchdog event or short-circuit detector event
- Extremes detector:
  - store minimum and maximum values of output data values
  - refreshed by software
- DMA may be used to read the conversion data
- Interrupts: end of conversion, overrun, analog watchdog, short-circuit, channel clock absence
- “regular” or “injected” conversions:
  - “regular” conversions can be requested at any time or even in continuous mode without having any impact on the timing of “injected” conversions

## 27.3 DFSDM implementation

This section describes the configuration implemented in DFSDMx.

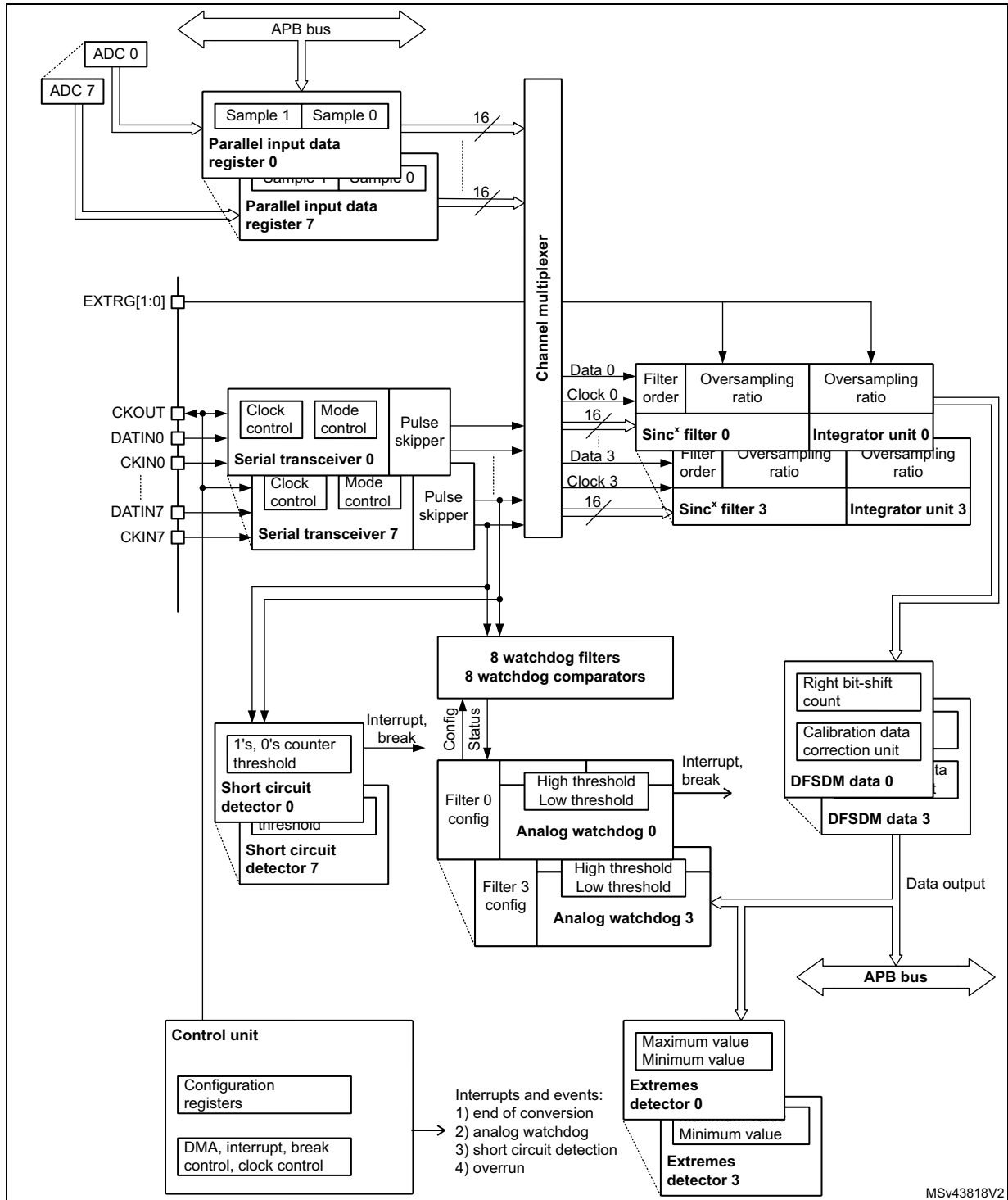
**Table 161. DFSDM1 implementation**

<b>DFSDM features</b>	<b>DFSDM1</b>
Number of channels	8
Number of filters	4
Input from internal ADC	X
Supported trigger sources	32
Pulses skipper	X
ID registers support	-

## 27.4 DFSDM functional description

### 27.4.1 DFSDM block diagram

Figure 163. Single DFSDM block diagram



1. This example shows 4 DFSDM filters and 8 input channels (max. configuration).

27.4.2 DFSDM pins and internal signals

Table 162. DFSDM external pins

Name	Signal Type	Remarks
VDD	Power supply	Digital power supply.
VSS	Power supply	Digital ground power supply.
CKIN[7:0]	Clock input	Clock signal provided from external $\Sigma\Delta$ modulator. FT input.
DATIN[7:0]	Data input	Data signal provided from external $\Sigma\Delta$ modulator. FT input.
CKOUT	Clock output	Clock output to provide clock signal into external $\Sigma\Delta$ modulator.
EXTRG[1:0]	External trigger signal	Input trigger from two EXTI signals to start analog conversion (from GPIOs: EXTI11, EXTI15).

Table 163. DFSDM internal signals

Name	Signal Type	Remarks
dfsdm_jtrg[31:0]	Internal/external trigger signal	Input trigger from internal/external trigger sources in order to start analog conversion (from internal sources: synchronous input, from external sources: asynchronous input with synchronization). See <a href="#">Table 164</a> for details.
dfsdm_break[3:0]	break signal output	Break signals event generation from Analog watchdog or short-circuit detector
dfsdm_dma[3:0]	DMA request signal	DMA request signal from each DFSDM_FLTx (x=0..3): end of injected conversion event.
dfsdm_it[3:0]	Interrupt request signal	Interrupt signal for each DFSDM_FLTx (x=0..3)
dfsdm_dat_adc[15:0]	ADC input data	Up to 4 internal ADC data buses as parallel inputs.

Table 164. DFSDM triggers connection

Trigger name	Trigger source
dfsdm_jtrg0	TIM1_TRGO
dfsdm_jtrg1	TIM1_TRGO2
dfsdm_jtrg2	TIM8_TRGO
dfsdm_jtrg3	TIM8_TRGO2
dfsdm_jtrg4	TIM3_TRGO
dfsdm_jtrg5	TIM4_TRGO
dfsdm_jtrg6	TIM16_OC1
dfsdm_jtrg7	TIM6_TRGO
dfsdm_jtrg8	TIM7_TRGO
dfsdm_jtrg[23:9]	Reserved

Table 164. DFSDM triggers connection (continued)

Trigger name	Trigger source
dfsdm_jtrg24	EXTI11
dfsdm_jtrg25	EXTI15
dfsdm_jtrg26	LTIMER1
dfsdm_jtrg[31:27]	Reserved

Table 165. DFSDM break connection

Break name	Break destination
dfsdm_break[0]	TIM1/TIM15 break
dfsdm_break[1]	TIM1 break2 / TIM16 break
dfsdm_break[2]	TIM8/TIM17 break
dfsdm_break[3]	TIM8 break2

### 27.4.3 DFSDM reset and clocks

#### DFSDM on-off control

The DFSDM interface is globally enabled by setting DFSDMEN=1 in the DFSDM\_CH0CFGR1 register. Once DFSDM is globally enabled, all input channels (y=0..7) and digital filters DFSDM\_FLTx (x=0..3) start to work if their enable bits are set (channel enable bit CHEN in DFSDM\_CHyCFGR1 and DFSDM\_FLTx enable bit DFEN in DFSDM\_FLTxCR1).

Digital filter x DFSDM\_FLTx (x=0..3) is enabled by setting DFEN=1 in the DFSDM\_FLTxCR1 register. Once DFSDM\_FLTx is enabled (DFEN=1), both Sinc<sup>x</sup> digital filter unit and integrator unit are reinitialized.

By clearing DFEN, any conversion which may be in progress is immediately stopped and DFSDM\_FLTx is put into stop mode. All register settings remain unchanged except DFSDM\_FLTxAWSR and DFSDM\_FLTxISR (which are reset).

Channel y (y=0..7) is enabled by setting CHEN=1 in the DFSDM\_CHyCFGR1 register. Once the channel is enabled, it receives serial data from the external  $\Sigma\Delta$  modulator or parallel internal data sources (ADCs or CPU/DMA wire from memory).

DFSDM must be globally disabled (by DFSDMEN=0 in DFSDM\_CH0CFGR1) before stopping the system clock to enter in the STOP mode of the device.

#### DFSDM clocks

The internal DFSDM clock  $f_{DFSDMCLK}$ , which is used to drive the channel transceivers, digital processing blocks (digital filter, integrator) and next additional blocks (analog watchdog, short-circuit detector, extremes detector, control block) is generated by the RCC block and is derived from the system clock SYSCLK or peripheral clock PCLK2 (see DFSDMSEL bit description in [Section 6.4.32: Peripherals independent clock configuration register \(RCC\\_CCIPR2\)](#)). The DFSDM clock is automatically stopped in stop mode (if DFEN = 0 for all DFSDM\_FLTx, x=0..3).



The DFSDM serial channel transceivers can receive an external serial clock to sample an external serial data stream. The internal DFSDM clock must be at least 4 times faster than the external serial clock if standard SPI coding is used, and 6 times faster than the external serial clock if Manchester coding is used.

DFSDM can provide one external output clock signal to drive external  $\Sigma\Delta$  modulator(s) clock input(s). It is provided on CKOUT pin. This output clock signal must be in the range specified in given device datasheet and is derived from DFSDM clock or from audio clock (see CKOUTSRC bit in DFSDM\_CH0CFGR1 register) by programmable divider in the range 2 - 256 (CKOUTDIV in DFSDM\_CH0CFGR1 register). Audio clock source is SAI1 clock selected by SAI1SEL[1:0] field in RCC configuration (see [Section 6.4.32: Peripherals independent clock configuration register \(RCC\\_CCIPR2\)](#)).

#### 27.4.4 Serial channel transceivers

There are 8 multiplexed serial data channels which can be selected for conversion by each filter or Analog watchdog or Short-circuit detector. Those serial transceivers receive data stream from external  $\Sigma\Delta$  modulator. Data stream can be sent in SPI format or Manchester coded format (see SITP[1:0] bits in DFSDM\_CHyCFGR1 register). The channel is enabled for operation by setting CHEN=1 in DFSDM\_CHyCFGR1 register.

##### Channel inputs selection

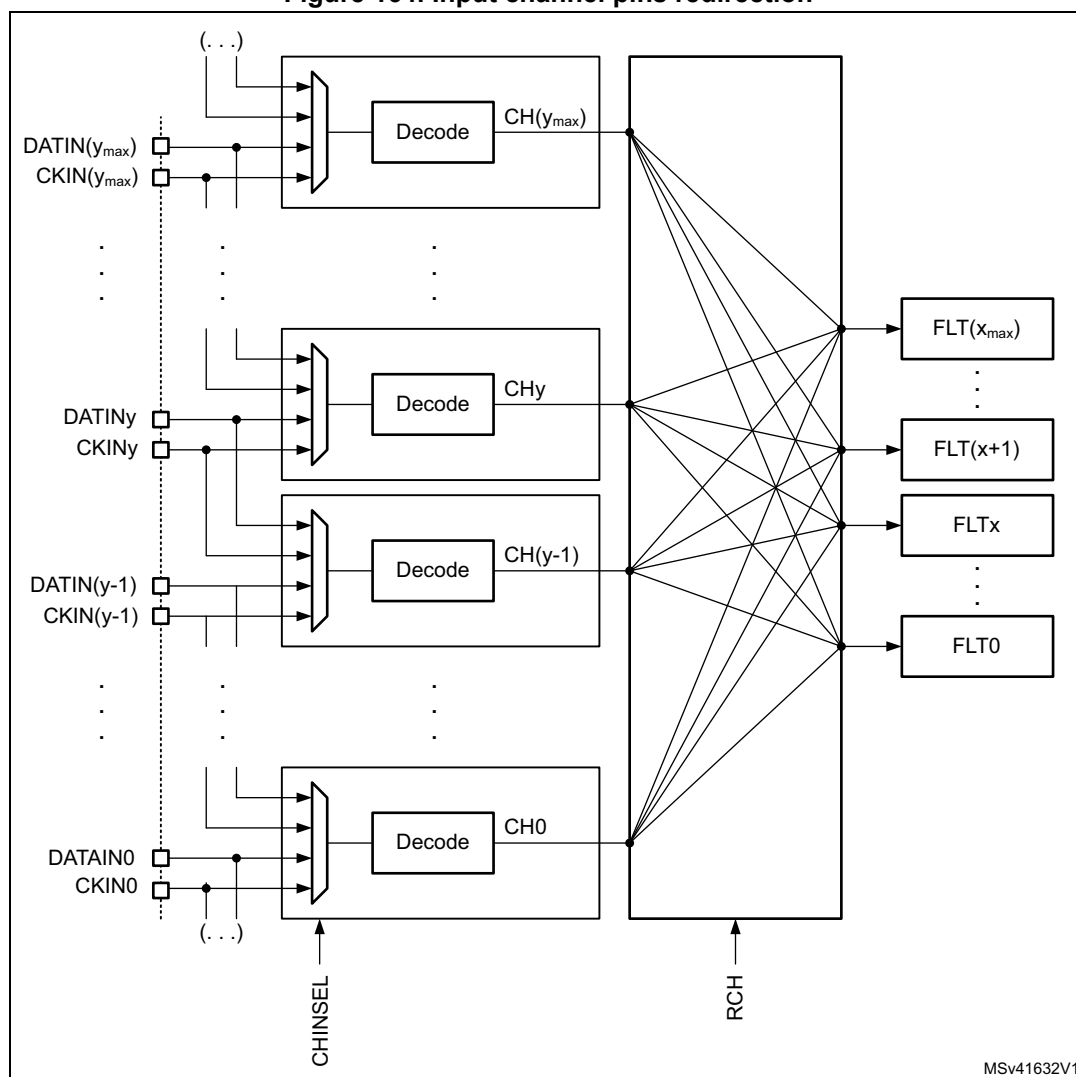
Serial inputs (data and clock signals) from DATINy and CKINy pins can be redirected from the following channel pins. This serial input channel redirection is set by CHINSEL bit in DFSDM\_CHyCFGR1 register.

Channel redirection can be used to collect audio data from PDM (pulse density modulation) stereo microphone type. PDM stereo microphone has one data and one clock signal. Data signal provides information for both left and right audio channel (rising clock edge samples for left channel and falling clock edge samples for right channel).

Configuration of serial channels for PDM microphone input:

- PDM microphone signals (data, clock) will be connected to DFSDM input serial channel y (DATINy, CKOUT) pins.
- Channel y will be configured: CHINSEL = 0 (input from given channel pins: DATINy, CKINy).
- Channel (y-1) (modulo 8) will be configured: CHINSEL = 1 (input from the following channel ((y-1)+1) pins: DATINy, CKINy).
- Channel y: SITP[1:0] = 0 (rising edge to strobe data) => left audio channel on channel y.
- Channel (y-1): SITP[1:0] = 1 (falling edge to strobe data) => right audio channel on channel y-1.
- Two DFSDM filters will be assigned to channel y and channel (y-1) (to filter left and right channels from PDM microphone).

Figure 164. Input channel pins redirection



**Output clock generation**

A clock signal can be provided on CKOUT pin to drive external  $\Sigma\Delta$  modulator clock inputs. The frequency of this CKOUT signal is derived from DFSDM clock or from audio clock (see CKOUTSRC bit in DFSDM\_CH0CFGR1 register) divided by a predivider (see CKOUTDIV bits in DFSDM\_CH0CFGR1 register). If the output clock is stopped, then CKOUT signal is set to low state (output clock can be stopped by CKOUTDIV=0 in DFSDM\_CHyCFGR1 register or by DFSDMEN=0 in DFSDM\_CH0CFGR1 register). The output clock stopping is performed:

- 4 system clocks after DFSDMEN is cleared (if CKOUTSRC=0)
- 1 system clock and 3 audio clocks after DFSDMEN is cleared (if CKOUTSRC=1)

Before changing CKOUTSRC the software has to wait for CKOUT being stopped to avoid glitch on CKOUT pin. The output clock signal frequency must be in the range 0 - 20 MHz.

### SPI data input format operation

In SPI format, the data stream is sent in serial format through data and clock signals. Data signal is always provided from DATINy pin. A clock signal can be provided externally from CKINy pin or internally from a signal derived from the CKOUT signal source.

In case of external clock source selection (SPICKSEL[1:0]=0) data signal (on DATINy pin) is sampled on rising or falling clock edge (of CKINy pin) according to SITP[1:0] bits setting (in DFSDM\_CHyCFGR1 register).

Internal clock sources - see SPICKSEL[1:0] in DFSDM\_CHyCFGR1 register:

- CKOUT signal:
  - For connection to external  $\Sigma\Delta$  modulator which uses directly its clock input (from CKOUT) to generate its output serial communication clock.
  - Sampling point: on rising/falling edge according to SITP[1:0] setting.
- CKOUT/2 signal (generated on CKOUT rising edge):
  - For connection to external  $\Sigma\Delta$  modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input rising edge).
  - Sampling point: on each second CKOUT falling edge.
- CKOUT/2 signal (generated on CKOUT falling edge):
  - For connection to external  $\Sigma\Delta$  modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input falling edge).
  - Sampling point: on each second CKOUT rising edge.

*Note:* An internal clock source can only be used when the external  $\Sigma\Delta$  modulator uses CKOUT signal as a clock input (to have synchronous clock and data operation).

Internal clock source usage can save CKINy pin connection (CKINy pins can be used for other purpose).

The clock source signal frequency must be in the range 0 - 20 MHz for SPI coding and less than  $f_{DFSDMCLK}/4$ .

### Manchester coded data input format operation

In Manchester coded format, the data stream is sent in serial format through DATINy pin only. Decoded data and clock signal are recovered from serial stream after Manchester decoding. There are two possible settings of Manchester codings (see SITP[1:0] bits in DFSDM\_CHyCFGR1 register):

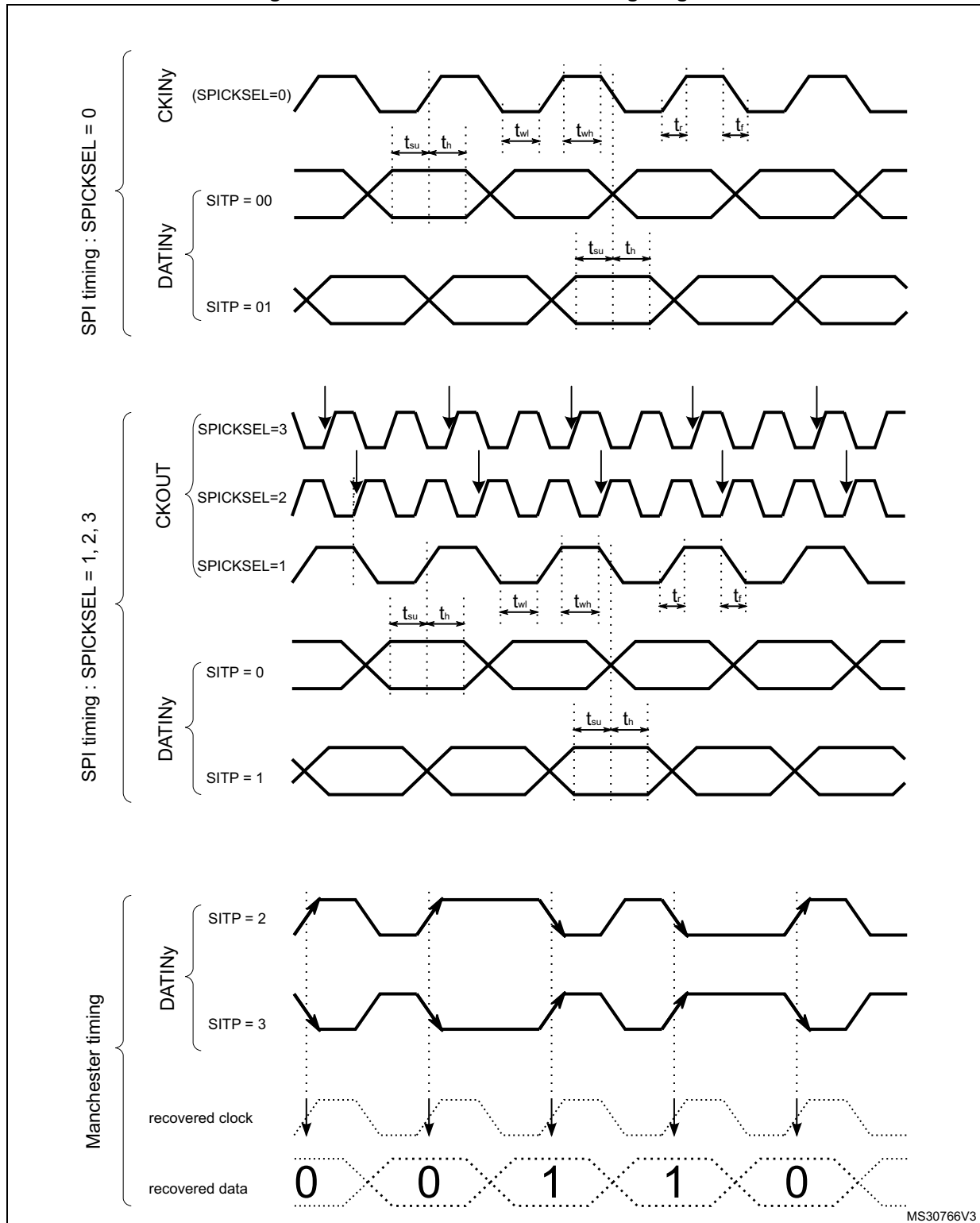
- signal rising edge = log 0; signal falling edge = log 1
- signal rising edge = log 1; signal falling edge = log 0

The recovered clock signal frequency for Manchester coding must be in the range 0 - 10 MHz and less than  $f_{DFSDMCLK}/6$ .

To correctly receive Manchester coded data, the CKOUTDIV divider (in DFSDM\_CH0CFGR1 register) must be set with respect to expected Manchester data rate according formula:

$$((CKOUTDIV + 1) \times T_{SYSCLK}) < T_{Manchester\ clock} < (2 \times CKOUTDIV \times T_{SYSCLK})$$

Figure 165. Channel transceiver timing diagrams



**Clock absence detection**

Channels serial clock inputs can be checked for clock absence/presence to ensure the correct operation of conversion and error reporting. Clock absence detection can be enabled or disabled on each input channel y by bit CKABEN in DFSDM\_CHyCFGR1 register. If enabled, then this clock absence detection is performed continuously on a given channel. A clock absence flag is set (CKABF[y] = 1) and an interrupt can be invoked (if CKABIE=1) in case of an input clock error (see CKABF[7:0] in DFSDM\_FLT0ISR register and CKABEN in DFSDM\_CHyCFGR1). After a clock absence flag clearing (by CLRCKABF in DFSDM\_FLT0ICR register), the clock absence flag is refreshed. Clock absence status bit CKABF[y] is set also by hardware when corresponding channel y is disabled (if CHEN[y] = 0 then CKABF[y] is held in set state).

When a clock absence event has occurred, the data conversion (and/or analog watchdog and short-circuit detector) provides incorrect data. The user should manage this event and discard given data while a clock absence is reported.

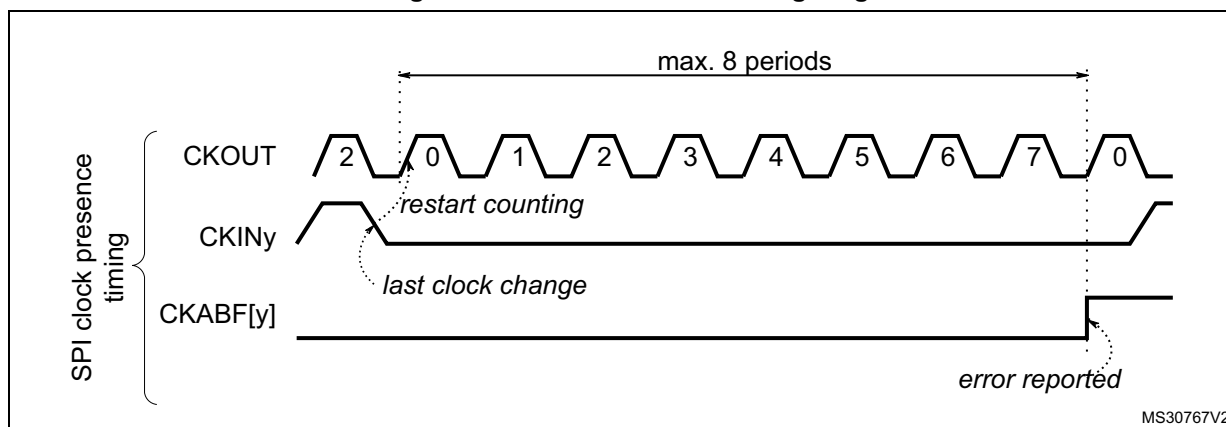
The clock absence feature is available only when the system clock is used for the CKOUT signal (CKOUTSRC=0 in DFSDM\_CH0CFGR1 register).

When the transceiver is not yet synchronized, the clock absence flag is set and cannot be cleared by CLRCKABF[y] bit (in DFSDM\_FLT0ICR register). The software sequence concerning clock absence detection feature should be:

- Enable given channel by CHEN = 1
- Try to clear the clock absence flag (by CLRCKABF = 1) until the clock absence flag is really cleared (CKABF = 0). At this time, the transceiver is synchronized (signal clock is valid) and is able to receive data.
- Enable the clock absence feature CKABEN = 1 and the associated interrupt CKABIE = 1 to detect if the SPI clock is lost or Manchester data edges are missing.

If SPI data format is used, then the clock absence detection is based on the comparison of an external input clock with an output clock generation (CKOUT signal). The external input clock signal into the input channel must be changed at least once per 8 signal periods of CKOUT signal (which is controlled by CKOUTDIV field in DFSDM\_CH0CFGR1 register).

**Figure 166. Clock absence timing diagram for SPI**



If Manchester data format is used, then the clock absence means that the clock recovery is unable to perform from Manchester coded signal. For a correct clock recovery, it is first necessary to receive data with 1 to 0 or 0 to 1 transition (see [Figure 168](#) for Manchester synchronization).

The detection of a clock absence in Manchester coding (after a first successful synchronization) is based on changes comparison of coded serial data input signal with output clock generation (CKOUT signal). There must be a voltage level change on DATINy pin during 2 periods of CKOUT signal (which is controlled by CKOUTDIV bits in DFSDM\_CH0CFGR1 register). This condition also defines the minimum data rate to be able to correctly recover the Manchester coded data and clock signals.

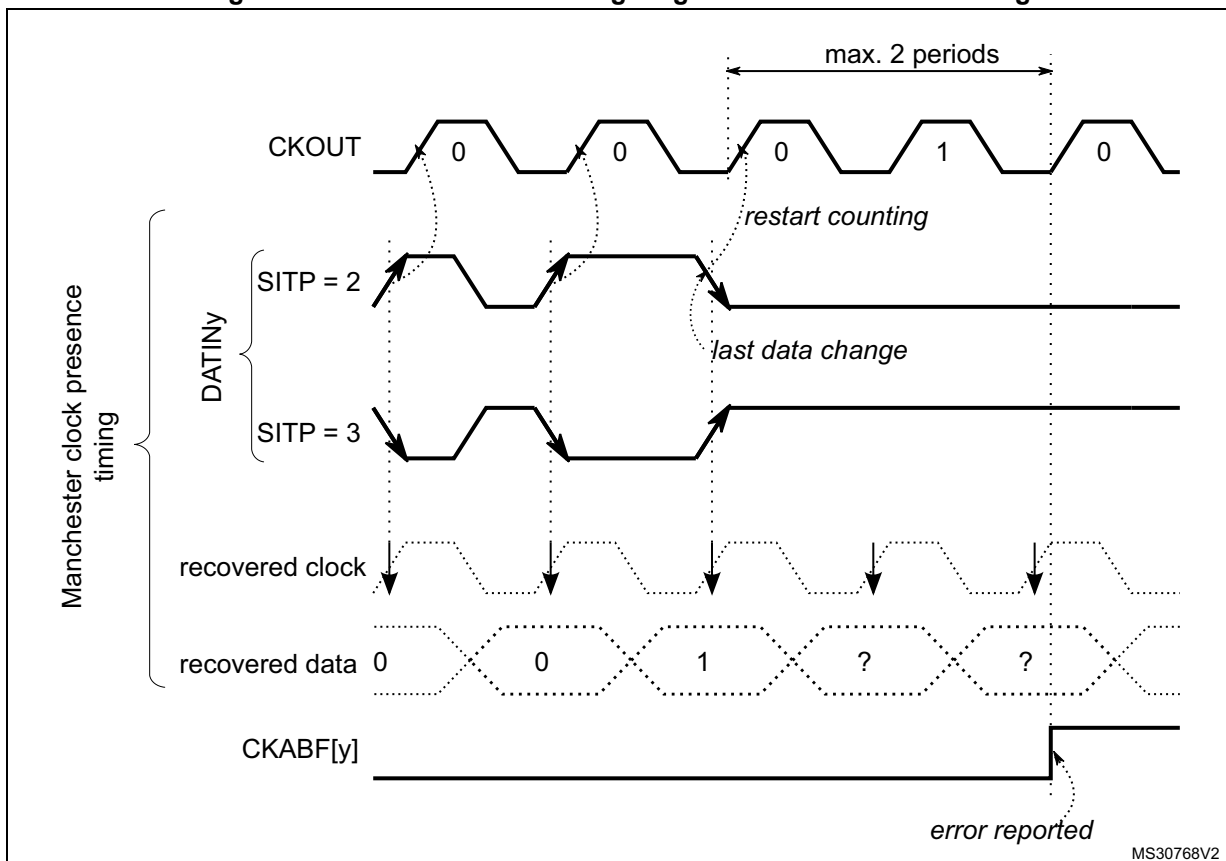
The maximum data rate of Manchester coded data must be less than the CKOUT signal.

So to correctly receive Manchester coded data, the CKOUTDIV divider must be set according the formula:

$$((CKOUTDIV + 1) \times T_{SYSCLK}) < T_{Manchester\ clock} < (2 \times CKOUTDIV \times T_{SYSCLK})$$

A clock absence flag is set (CKABF[y] = 1) and an interrupt can be invoked (if CKABIE=1) in case of an input clock recovery error (see CKABF[7:0] in DFSDM\_FLT0ISR register and CKABEN in DFSDM\_CHyCFGR1). After a clock absence flag clearing (by CLRCKABF in DFSDM\_FLT0ICR register), the clock absence flag is refreshed.

Figure 167. Clock absence timing diagram for Manchester coding



### Manchester/SPI code synchronization

The Manchester coded stream must be synchronized the first time after enabling the channel (CHEN=1 in DFSDM\_CHyCFGR1 register). The synchronization ends when a data transition from 0 to 1 or from 1 to 0 (to be able to detect valid data edge) is received. The end of the synchronization can be checked by polling CKABF[y]=0 for a given channel after it has been cleared by CLRCKABF[y] in DFSDM\_FLT0ICR, following the software sequence detailed hereafter:

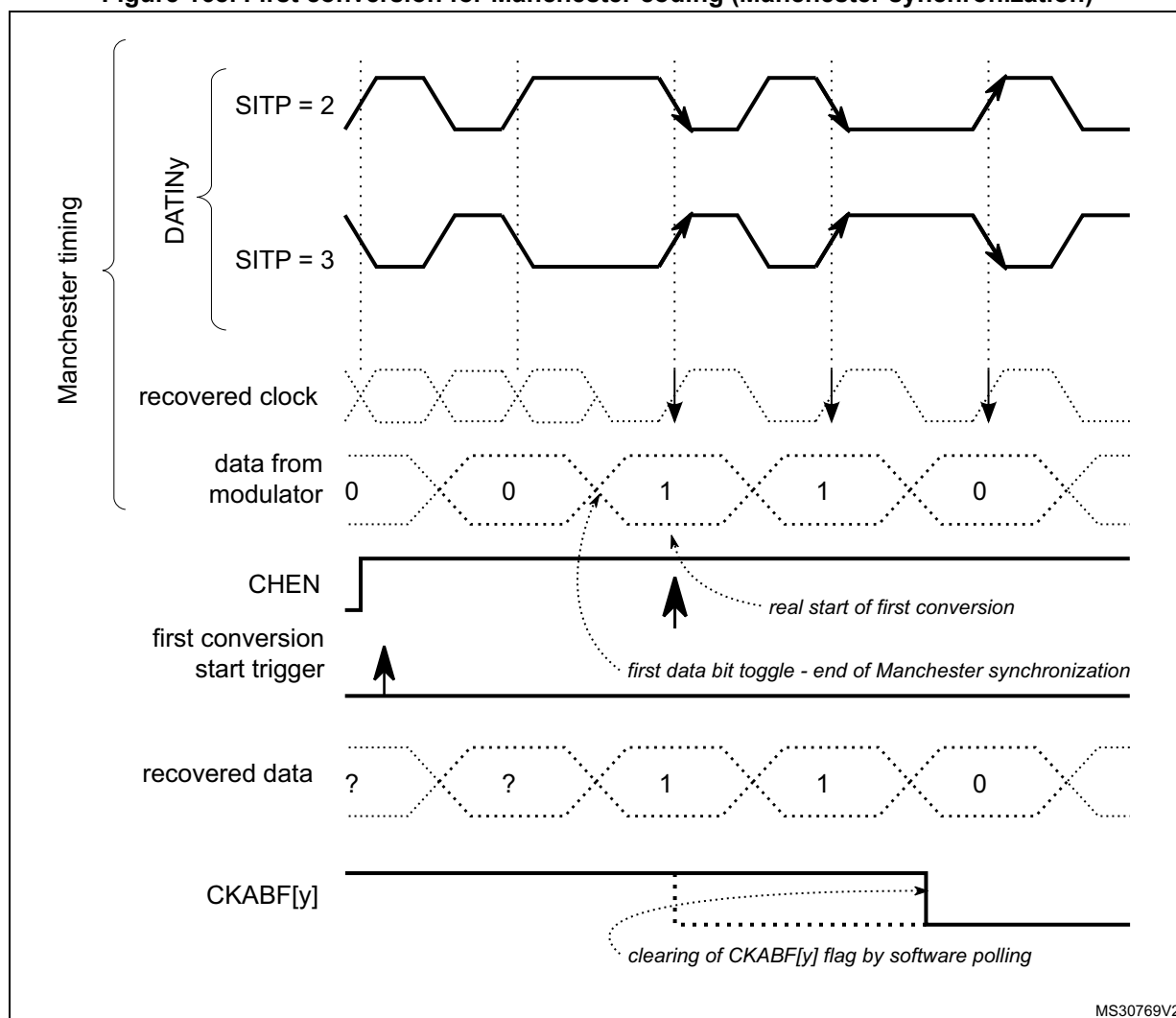
CKABF[y] flag is cleared by setting CLRCKABF[y] bit. If channel y is not yet synchronized the hardware immediately set the CKABF[y] flag. Software is then reading back the CKABF[y] flag and if it is set then perform again clearing of this flag by setting CLRCKABF[y] bit. This software sequence (polling of CKABF[y] flag) continues until CKABF[y] flag is set (signaling that Manchester stream is synchronized). To be able to synchronize/receive Manchester coded data the CKOUTDIV divider (in DFSDM\_CH0CFGR1 register) must be set with respect to expected Manchester data rate according the formula below.

$$((CKOUTDIV + 1) \times T_{SYSCLK}) < T_{Manchester\ clock} < (2 \times CKOUTDIV \times T_{SYSCLK})$$

SPI coded stream is synchronized after first detection of clock input signal (valid rising/falling edge).

*Note:* When the transceiver is not yet synchronized, the clock absence flag is set and cannot be cleared by CLRCKABF[y] bit (in DFSDM\_FLT0ICR register).

Figure 168. First conversion for Manchester coding (Manchester synchronization)



### External serial clock frequency measurement

The measuring of a channel serial clock input frequency provides a real data rate from an external  $\Sigma\Delta$  modulator, which is important for application purposes.

An external serial clock input frequency can be measured by a timer counting DFSDM clocks ( $f_{DFSDMCLK}$ ) during one conversion duration. The counting starts at the first input data clock after a conversion trigger (regular or injected) and finishes by last input data clock before conversion ends (end of conversion flag is set). Each conversion duration (time between first serial sample and last serial sample) is updated in counter CNVCNT[27:0] in register DFSDM\_FLTxCNVTIMR when the conversion finishes (JEOCF=1 or REOCF=1). The user can then compute the data rate according to the digital filter settings (FORD, FOSR, IOSR, FAST). The external serial frequency measurement is stopped only if the filter is bypassed (FOSR=0, only integrator is active, CNVCNT[27:0]=0 in DFSDM\_FLTxCNVTIMR register).

In case of parallel data input ([Section 27.4.6: Parallel data inputs](#)) the measured frequency is the average input data rate during one conversion.



*Note:* When conversion is interrupted (e.g. by disabling/enabling the selected channel) the interruption time is also counted in CNVCNT[27:0]. Therefore it is recommended to not interrupt the conversion for correct conversion duration result.

Conversion times:

**injected conversion or regular conversion with FAST = 0 (or first conversion if FAST=1):**

for Sinc<sup>x</sup> filters (x=1..5):

$$t = \text{CNVCNT} / f_{\text{DFSDMCLK}} = [F_{\text{OSR}} * (I_{\text{OSR}} - 1 + F_{\text{ORD}}) + F_{\text{ORD}}] / f_{\text{CKIN}}$$

for FastSinc filter:

$$t = \text{CNVCNT} / f_{\text{DFSDMCLK}} = [F_{\text{OSR}} * (I_{\text{OSR}} - 1 + 4) + 2] / f_{\text{CKIN}}$$

**regular conversion with FAST = 1 (except first conversion):**

for Sinc<sup>x</sup> and FastSinc filters:

$$t = \text{CNVCNT} / f_{\text{DFSDMCLK}} = [F_{\text{OSR}} * I_{\text{OSR}}] / f_{\text{CKIN}}$$

**in case if F<sub>OSR</sub> = FOSR[9:0]+1 = 1 (filter bypassed, active only integrator):**

$$t = I_{\text{OSR}} / f_{\text{CKIN}} \text{ (... but CNVCNT=0)}$$

where:

- $f_{\text{CKIN}}$  is the channel input clock frequency (on given channel CKINy pin) or input data rate (in case of parallel data input)
- $F_{\text{OSR}}$  is the filter oversampling ratio:  $F_{\text{OSR}} = \text{FOSR}[9:0] + 1$  (see DFSDM\_FLTxFCR register)
- $I_{\text{OSR}}$  is the integrator oversampling ratio:  $I_{\text{OSR}} = \text{IOSR}[7:0] + 1$  (see DFSDM\_FLTxFCR register)
- $F_{\text{ORD}}$  is the filter order:  $F_{\text{ORD}} = \text{FORD}[2:0]$  (see DFSDM\_FLTxFCR register)

### Channel offset setting

Each channel has its own offset setting (in register) which is finally subtracted from each conversion result (injected or regular) from a given channel. Offset correction is performed after the data right bit shift. The offset is stored as a 24-bit signed value in OFFSET[23:0] field in DFSDM\_CHyCFGR2 register.

### Data right bit shift

To have the result aligned to a 24-bit value, each channel defines a number of right bit shifts which will be applied on each conversion result (injected or regular) from a given channel. The data bit shift number is stored in DTRBS[4:0] bits in DFSDM\_CHyCFGR2 register.

The right bit-shift is rounding the result to nearest integer value. The sign of shifted result is maintained, in order to have valid 24-bit signed format of result data.

### Pulses skipper

Purpose of the pulses skipper is to implement delay line like behavior for given input channel(s). Given number of samples from input serial data stream (serial stream only) can be discarded before they enter into the filter. This data discarding is performed by skipping given number of sampling input clock pulses (given serial data samples are then not sampled by filter). The sampling clock is gated by pulses skipper function for given number of clock pulses. When given clock pulses are skipped then the filtering continues for following input data. With comparison to non skipped data stream this operation causes that

the final output sample (and next samples) from filter will be calculated from later input data. This final sample then looks a bit in forward - because it is calculated from newer input samples than the “non-skipped” sample. The final “skipped sample” is converted later because the skipped input data samples must be replaced by followed input data samples. The final data buffers behavior (skipped and non-skipped output data buffers comparison) looks like the non-skipped data stream is a bit delayed - both data buffers will be phase shifted.

Number of clock pulses to be skipped should be written into PLSSKP[5:0] field in DFSDM\_CHyDLYR register. Once PLSSKP[5:0] field is written the execution of pulses skipping is started on given channel. PLSSKP[5:0] field can be read in order to check the progress of pulses skipper. When PLSSKP[5:0]=0 means that pulses skipping has been executed.

Up to 63 clock pulses can be skip with a single write operation into PLSSKP[5:0]. If more pulses need to be skipped, then user has to write several times into the PLSSKP[5:0] field. The application software should handle cumulative skipped clock number per each filter.

### 27.4.5 Configuring the input serial interface

The following parameters must be configured for the input serial interface:

- **Output clock predivider.** There is a programmable predivider to generate the output clock from DFSDM clock (2 - 256). It is defined by CKOUTDIV[7:0] bits in DFSDM\_CH0CFGR1 register.
- **Serial interface type and input clock phase.** Selection of SPI or Manchester coding and sampling edge of input clock. It is defined by SITP [1:0] bits in DFSDM\_CHyCFGR1 register.
- **Input clock source.** External source from CKINy pin or internal from CKOUT pin. It is defined by SPICKSEL[1:0] field in DFSDM\_CHyCFGR1 register.
- **Final data right bit-shift.** Defines the final data right bit shift to have the result aligned to a 24-bit value. It is defined by DTRBS[4:0] in DFSDM\_CHyCFGR2 register.
- **Channel offset per channel.** Defines the analog offset of a given serial channel (offset of connected external  $\Sigma\Delta$  modulator). It is defined by OFFSET[23:0] bits in DFSDM\_CHyCFGR2 register.
- **short-circuit detector and clock absence per channel enable.** To enable or disable the short-circuit detector (by SCDEN bit) and the clock absence monitoring (by CKABEN bit) on a given serial channel in register DFSDM\_CHyCFGR1.
- **Analog watchdog filter and short-circuit detector threshold settings.** To configure channel analog watchdog filter parameters and channel short-circuit detector parameters. Configurations are defined in DFSDM\_CHyAWSCDR register.

### 27.4.6 Parallel data inputs

Each input channel provides a register for 16-bit parallel data input (besides serial data input). Each 16-bit parallel input can be sourced from internal data sources only:

- internal ADC results
- direct CPU/DMA writing.

The selection for using serial or parallel data input for a given channel is done by field DATMPX[1:0] of DFSDM\_CHyCFGR1 register. In DATMPX[1:0] is also defined the parallel data source: internal ADC or direct write by CPU/DMA.

Each channel contains a 32-bit data input register DFSDM\_CHyDATINR in which it can be written a 16-bit data. Data are in 16-bit signed format. Those data can be used as input to the digital filter which is accepting 16-bit parallel data.

If serial data input is selected (DATMPX[1:0] = 0), the DFSDM\_CHyDATINR register is write protected.

### Input from internal ADC

In case of ADC data parallel input (DATMPX[1:0]=1) the ADC[y+1] result is assigned to channel y input (ADC1 is filling DFSDM\_CHDATIN0R register, ADC2 is filling DFSDM\_CHDATIN1R register, ... , ADC8 is filling DFSDM\_CHDATIN7R register). End of conversion event from ADC[y+1] causes update of channel y data (parallel data from ADC[y+1] are put as next sample to digital filter). Data from ADC[y+1] is written into DFSDM\_CHyDATINR register (field INDAT0[15:0]) when end of conversion event occurred.

The setting of data packing mode (DATPACK[1:0] in the DFSDM\_CHyCFGR1 register) has no effect in case of ADC data input.

### Input from memory (direct CPU/DMA write)

The direct data write into DFSDM\_CHyDATINR register by CPU or DMA (DATMPX[1:0]=2) can be used as data input in order to process digital data streams from memory or peripherals.

Data can be written by CPU or DMA into DFSDM\_CHyDATINR register:

#### 1. CPU data write:

Input data are written directly by CPU into DFSDM\_CHyDATINR register.

#### 2. DMA data write:

The DMA should be configured in memory-to-memory transfer mode to transfer data from memory buffer into DFSDM\_CHyDATINR register. The destination memory address is the address of DFSDM\_CHyDATINR register. Data are transferred at DMA transfer speed from memory to DFSDM parallel input.

This DMA transfer is different from DMA used to read DFSDM conversion results. Both DMA can be used at the same time - first DMA (configured as memory-to-memory transfer) for input data writings and second DMA (configured as peripheral-to-memory transfer) for data results reading.

The accesses to DFSDM\_CHyDATINR can be either 16-bit or 32-bit wide, allowing to load respectively one or two samples in one write operation. 32-bit input data register (DFSDM\_CHyDATINR) can be filled with one or two 16-bit data samples, depending on the data packing operation mode defined in field DATPACK[1:0] of DFSDM\_CHyCFGR1 register:

#### 1. Standard mode (DATPACK[1:0]=0):

Only one sample is stored in field INDAT0[15:0] of DFSDM\_CHyDATINR register which is used as input data for channel y. The upper 16 bits (INDAT1[15:0]) are ignored and write protected. The digital filter must perform one input sampling (from INDAT0[15:0]) to empty data register after it has been filled by CPU/DMA. This mode is used together with 16-bit CPU/DMA access to DFSDM\_CHyDATINR register to load one sample per write operation.

#### 2. Interleaved mode (DATPACK[1:0]=1):

DFSDM\_CHyDATINR register is used as a two sample buffer. The first sample is stored in INDAT0[15:0] and the second sample is stored in INDAT1[15:0]. The digital

filter must perform two input samplings from channel y to empty DFSDM\_CHyDATINR register. This mode is used together with 32-bit CPU/DMA access to DFSDM\_CHyDATINR register to load two samples per write operation.

**3. Dual mode (DATPACK[1:0]=2):**

Two samples are written into DFSDM\_CHyDATINR register. The data INDAT0[15:0] is for channel y, the data in INDAT1[15:0] is for channel y+1. The data in INDAT1[15:0] is automatically copied INDAT0[15:0] of the following (y+1) channel data register DFSDM\_CH[y+1]DATINR). The digital filters must perform two samplings - one from channel y and one from channel (y+1) - in order to empty DFSDM\_CHyDATINR registers.

Dual mode setting (DATPACK[1:0]=2) is available only on even channel numbers (y = 0, 2, 4, 6). If odd channel (y = 1, 3, 5, 7) is set to Dual mode then both INDAT0[15:0] and INDAT1[15:0] parts are write protected for this channel. If even channel is set to Dual mode then the following odd channel must be set into Standard mode (DATPACK[1:0]=0) for correct cooperation with even channels.

See [Figure 169](#) for DFSDM\_CHyDATINR registers data modes and assignments of data samples to channels.

**Figure 169. DFSDM\_CHyDATINR registers operation modes and assignment**

Standard mode		Interleaved mode		Dual mode		
31	16 15 0	31	16 15 0	31	16 15 0	
Unused	Ch0 (sample 0)	Ch0 (sample 1) Ch0 (sample 0)	Ch1 (sample 0) Ch0 (sample 0)	Ch1 (sample 0) Ch0 (sample 0)	Ch0 (sample 0)	<b>y = 0</b>
Unused	Ch1 (sample 0)	Ch1 (sample 1) Ch1 (sample 0)	Ch2 (sample 0) Ch1 (sample 0)	Unused	Ch1 (sample 0)	<b>y = 1</b>
Unused	Ch2 (sample 0)	Ch2 (sample 1) Ch2 (sample 0)	Ch3 (sample 0) Ch2 (sample 0)	Ch3 (sample 0) Ch2 (sample 0)	Ch2 (sample 0)	<b>y = 2</b>
Unused	Ch3 (sample 0)	Ch3 (sample 1) Ch3 (sample 0)	Ch4 (sample 0) Ch3 (sample 0)	Unused	Ch3 (sample 0)	<b>y = 3</b>
Unused	Ch4 (sample 0)	Ch4 (sample 1) Ch4 (sample 0)	Ch5 (sample 0) Ch4 (sample 0)	Ch5 (sample 0) Ch4 (sample 0)	Ch4 (sample 0)	<b>y = 4</b>
Unused	Ch5 (sample 0)	Ch5 (sample 1) Ch5 (sample 0)	Ch6 (sample 0) Ch5 (sample 0)	Unused	Ch5 (sample 0)	<b>y = 5</b>
Unused	Ch6 (sample 0)	Ch6 (sample 1) Ch6 (sample 0)	Ch7 (sample 0) Ch6 (sample 0)	Ch7 (sample 0) Ch6 (sample 0)	Ch6 (sample 0)	<b>y = 6</b>
Unused	Ch7 (sample 0)	Ch7 (sample 1) Ch7 (sample 0)	Unused	Ch7 (sample 0)	Ch7 (sample 0)	<b>y = 7</b>

MS35354V3

The write into DFSDM\_CHyDATINR register to load one or two samples must be performed after the selected input channel (channel y) is enabled for data collection (starting conversion for channel y). Otherwise written data are lost for next processing.

For example: for single conversion and interleaved mode, do not start writing pair of data samples into DFSDM\_CHyDATINR before the single conversion is started (any data present in the DFSDM\_CHyDATINR before starting a conversion is discarded).

**27.4.7 Channel selection**

There are 8 multiplexed channels which can be selected for conversion using the injected channel group and/or using the regular channel.

The **injected channel group** is a selection of any or all of the 8 channels. JCHG[7:0] in the DFSDM\_FLTxJCHGR register selects the channels of the injected group, where JCHG[y]=1 means that channel y is selected.

Injected conversions can operate in scan mode (JSCAN=1) or single mode (JSCAN=0). In scan mode, each of the selected channels is converted, one after another. The lowest channel (channel 0, if selected) is converted first, followed immediately by the next higher channel until all the channels selected by JCHG[7:0] have been converted. In single mode (JSCAN=0), only one channel from the selected channels is converted, and the channel selection is moved to the next channel. Writing to JCHG[7:0] if JSCAN=0 resets the channel selection to the lowest selected channel.

Injected conversions can be launched by software or by a trigger. They are never interrupted by regular conversions.

The **regular channel** is a selection of just one of the 8 channels. RCH[2:0] in the DFSDM\_FLTxCR1 register indicates the selected channel.

Regular conversions can be launched only by software (not by a trigger). A sequence of continuous regular conversions is temporarily interrupted when an injected conversion is requested.

Performing a conversion on a disabled channel (CHEN=0 in DFSDM\_CHyCFGR1 register) causes that the conversion will never end - because no input data is provided (with no clock signal). In this case, it is necessary to enable a given channel (CHEN=1 in DFSDM\_CHyCFGR1 register) or to stop the conversion by DFEN=0 in DFSDM\_FLTxCR1 register.

## 27.4.8 Digital filter configuration

DFSDM contains a Sinc<sup>x</sup> type digital filter implementation. This Sinc<sup>x</sup> filter performs an input digital data stream filtering, which results in decreasing the output data rate (decimation) and increasing the output data resolution. The Sinc<sup>x</sup> digital filter is configurable in order to reach the required output data rates and required output data resolution. The configurable parameters are:

- Filter order/type: (see FORD[2:0] bits in DFSDM\_FLTxFCR register):
  - FastSinc
  - Sinc<sup>1</sup>
  - Sinc<sup>2</sup>
  - Sinc<sup>3</sup>
  - Sinc<sup>4</sup>
  - Sinc<sup>5</sup>
- Filter oversampling/decimation ratio (see FOSR[9:0] bits in DFSDM\_FLTxFCR register):
  - FOSR = 1-1024 - for FastSinc filter and Sinc<sup>x</sup> filter x = F<sub>ORD</sub> = 1..3
  - FOSR = 1-215 - for Sinc<sup>x</sup> filter x = F<sub>ORD</sub> = 4
  - FOSR = 1-73 - for Sinc<sup>x</sup> filter x = F<sub>ORD</sub> = 5

The filter has the following transfer function (impulse response in H domain):

- Sinc<sup>x</sup> filter type: 
$$H(z) = \left( \frac{1 - z^{-FOSR}}{1 - z^{-1}} \right)^x$$
- FastSinc filter type: 
$$H(z) = \left( \frac{1 - z^{-FOSR}}{1 - z^{-1}} \right)^2 \cdot (1 + z^{-(2 \cdot FOSR)})$$

Figure 170. Example: Sinc<sup>3</sup> filter response

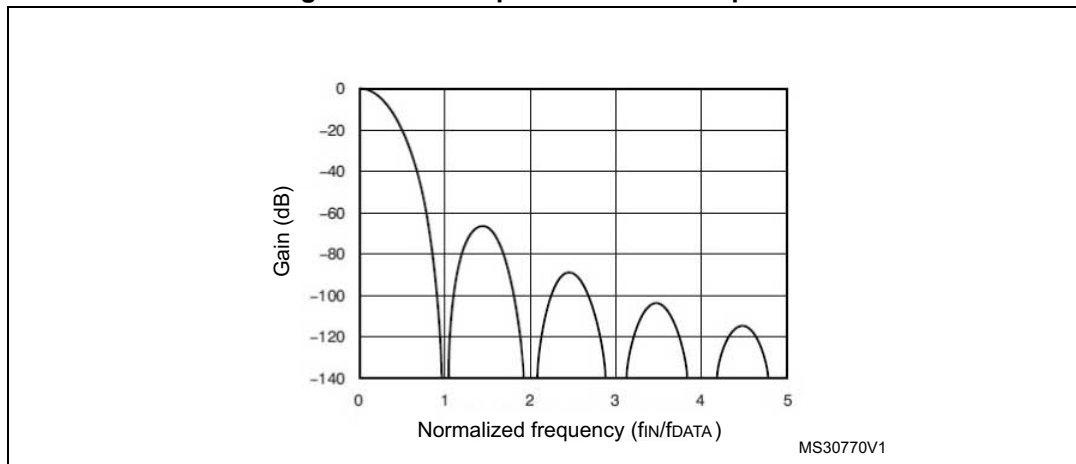


Table 166. Filter maximum output resolution (peak data values from filter output) for some FOSR values

FOSR	Sinc <sup>1</sup>	Sinc <sup>2</sup>	FastSinc	Sinc <sup>3</sup>	Sinc <sup>4</sup>	Sinc <sup>5</sup>
x	+/- x	+/- x <sup>2</sup>	+/- 2x <sup>2</sup>	+/- x <sup>3</sup>	+/- x <sup>4</sup>	+/- x <sup>5</sup>
4	+/- 4	+/- 16	+/- 32	+/- 64	+/- 256	+/- 1024
8	+/- 8	+/- 64	+/- 128	+/- 512	+/- 4096	-
32	+/- 32	+/- 1024	+/- 2048	+/- 32768	+/- 1048576	+/- 33554432
64	+/- 64	+/- 4096	+/- 8192	+/- 262144	+/- 16777216	+/- 1073741824
128	+/- 128	+/- 16384	+/- 32768	+/- 2097152	+/- 268435456	Result can overflow on full scale input (> 32-bit signed integer)
256	+/- 256	+/- 65536	+/- 131072	+/- 16777216		
1024	+/- 1024	+/- 1048576	+/- 2097152	+/- 1073741824		

For more information about Sinc filter type properties and usage, it is recommended to study the theory about digital filters (more resources can be downloaded from internet).

### 27.4.9 Integrator unit

The integrator performs additional decimation and a resolution increase of data coming from the digital filter. The integrator simply performs the sum of data from a digital filter for a given number of data samples from a filter.

The integrator oversampling ratio parameter defines how many data counts will be summed to one data output from the integrator. IOSR can be set in the range 1-256 (see IOSR[7:0] bits description in DFSDM\_FLTxFCR register).

**Table 167. Integrator maximum output resolution (peak data values from integrator output) for some IOSR values and FOSR = 256 and Sinc<sup>3</sup> filter type (largest data)**

IOSR	Sinc <sup>1</sup>	Sinc <sup>2</sup>	FastSinc	Sinc <sup>3</sup>	Sinc <sup>4</sup>	Sinc <sup>5</sup>
x	+/- FOSR. x	+/- FOSR <sup>2</sup> . x	+/- 2.FOSR <sup>2</sup> . x	+/- FOSR <sup>3</sup> . x	+/- FOSR <sup>4</sup> . x	+/- FOSR <sup>5</sup> . x
4	-	-	-	+/- 67 108 864	-	-
32	-	-	-	+/- 536 870 912	-	-
128	-	-	-	+/- 2 147 483 648	-	-
256	-	-	-	+/- 2 <sup>32</sup>	-	-

**27.4.10 Analog watchdog**

The analog watchdog purpose is to trigger an external signal (break or interrupt) when an analog signal reaches or crosses given maximum and minimum threshold values. An interrupt/event/break generation can then be invoked.

Each analog watchdog will supervise serial data receiver outputs (after the analog watchdog filter on each channel) or data output register (current injected or regular conversion result) according to AWFSEL bit setting (in DFSDM\_FLTxCR1 register). The input channels to be monitored or not by the analog watchdog x will be selected by AWDCH[7:0] in DFSDM\_FLTxCR2 register.

Analog watchdog conversions on input channels are independent from standard conversions. In this case, the analog watchdog uses its own filters and signal processing on each input channel independently from the main injected or regular conversions. Analog watchdog conversions are performed in a continuous mode on the selected input channels in order to watch channels also when main injected or regular conversions are paused (RCIP = 0, JCIP = 0).

There are high and low threshold registers which are compared with given data values (set by AWHT[23:0] bits in DFSDM\_FLTxAWHTR register and by AWLT[23:0] bits in DFSDM\_FLTxAWLTR register).



There are 2 options for comparing the threshold registers with the data values

- Option1: in this case, the input data are taken from final output data register (AWFSEL=0). This option is characterized by:
  - high input data resolution (up to 24-bits)
  - slow response time - inappropriate for fast response applications like overcurrent detection
  - for the comparison the final data are taken after bit shifting and offset data correction
  - final data are available only after main regular or injected conversions are performed
  - can be used in case of parallel input data source (DATMPX[1:0] ≠ 0 in DFSDM\_CHyCFGR1 register)
- Option2: in this case, the input data are taken from any serial data receivers output (AWFSEL=1). This option is characterized by:
  - input serial data are processed by dedicated analog watchdog Sinc<sup>x</sup> channel filters with configurable oversampling ratio (1..32) and filter order (1..3) (see AWFOSR[4:0] and AWFORD[1:0] bits setting in DFSDM\_CHyAWSCDR register)
  - lower resolution (up to 16-bit)
  - fast response time - appropriate for applications which require a fast response like overcurrent/overvoltage detection)
  - data are available in continuous mode independently from main regular or injected conversions activity

In case of input channels monitoring (AWFSEL=1), the data for comparison to threshold is taken from channels selected by AWDCH[7:0] field (DFSDM\_FLTxCR2 register). Each of the selected channels filter result is compared to one threshold value pair (AWHT[23:0] / AWLT[23:0]). In this case, only higher 16 bits (AWHT[23:8] / AWLT[23:8]) define the 16-bit threshold compared with the analog watchdog filter output because data coming from the analog watchdog filter is up to a 16-bit resolution. Bits AWHT[7:0] / AWLT[7:0] are not taken into comparison in this case (AWFSEL=1).

Parameters of the analog watchdog filter configuration for each input channel are set in DFSDM\_CHyAWSCDR register (filter order AWFORD[1:0] and filter oversampling ratio AWFOSR[4:0]).

Each input channel has its own comparator which compares the analog watchdog data (from analog watchdog filter) with analog watchdog threshold values (AWHT/AWLT). When several channels are selected (field AWDCH[7:0] field of DFSDM\_FLTxCR2 register), several comparison requests may be received simultaneously. In this case, the channel request with the lowest number is managed first and then continuing to higher selected channels. For each channel, the result can be recorded in a separate flag (fields AWHTF[7:0], AWLTF[7:0] of DFSDM\_FLTxAWSR register). Each channel request is executed in 8 DFSDM clock cycles. So, the bandwidth from each channel is limited to 8 DFSDM clock cycles (if AWDCH[7:0] = 0xFF). Because the maximum input channel sampling clock frequency is the DFSDM clock frequency divided by 4, the configuration AWFOSR = 0 (analog watchdog filter is bypassed) cannot be used for analog watchdog feature at this input clock speed. Therefore user must properly configure the number of watched channels and analog watchdog filter parameters with respect to input sampling clock speed and DFSDM frequency.



Analog watchdog filter data for given channel y is available for reading by firmware on field WDATA[15:0] in DFSDM\_CHyWDATR register. That analog watchdog filter data is converted continuously (if CHEN=1 in DFSDM\_CHyCFGR1 register) with the data rate given by the analog watchdog filter setting and the channel input clock frequency.

The analog watchdog filter conversion works like a regular Fast Continuous Conversion without the intergator. The number of serial samples needed for one result from analog watchdog filter output (at channel input clock frequency  $f_{CKIN}$ ):

first conversion:

for Sinc<sup>x</sup> filters (x=1..5): number of samples =  $[F_{OSR} * F_{ORD} + F_{ORD} + 1]$

for FastSinc filter: number of samples =  $[F_{OSR} * 4 + 2 + 1]$

next conversions:

for Sinc<sup>x</sup> and FastSinc filters: number of samples =  $[F_{OSR} * IOSR]$

where:

$F_{OSR}$  ..... filter oversampling ratio:  $F_{OSR} = AWFOSR[4:0] + 1$  (see DFSDM\_CHyAWSCDR register)

$F_{ORD}$  ..... the filter order:  $F_{ORD} = AWFORD[1:0]$  (see DFSDM\_CHyAWSCDR register)

In case of output data register monitoring (AWFSEL=0), the comparison is done after a right bit shift and an offset correction of final data (see OFFSET[23:0] and DTRBS[4:0] fields in DFSDM\_CHyCFGR2 register). A comparison is performed after each injected or regular end of conversion for the channels selected by AWDCH[7:0] field (in DFSDM\_FLTxCR2 register).

The status of an analog watchdog event is signaled in DFSDM\_FLTxAWSR register where a given event is latched. AWHTF[y]=1 flag signalizes crossing AWHT[23:0] value on channel y. AWLTF[y]=1 flag signalizes crossing AWLT[23:0] value on channel y. Latched events in DFSDM\_FLTxAWSR register are cleared by writing '1' into the corresponding clearing bit CLRAWHTF[y] or CLRAWLTF[y] in DFSDM\_FLTxAWCFR register.

The global status of an analog watchdog is signaled by the AWDF flag bit in DFSDM\_FLTxISR register (it is used for the fast detection of an interrupt source). AWDF=1 signalizes that at least one watchdog occurred (AWHTF[y]=1 or AWLTF[y]=1 for at least one channel). AWDF bit is cleared when all AWHTF[7:0] and AWLTF[7:0] are cleared.

An analog watchdog event can be assigned to break output signal. There are four break outputs to be assigned to a high or low threshold crossing event (dfsdm\_break[3:0]). The break signal assignment to a given analog watchdog event is done by BKAWH[3:0] and BKAWL[3:0] fields in DFSDM\_FLTxAWHTR and DFSDM\_FLTxAWLTR register.

### 27.4.11 Short-circuit detector

The purpose of a short-circuit detector is to signalize with a very fast response time if an analog signal reached saturated values (out of full scale ranges) and remained on this value given time. This behavior can detect short-circuit or open circuit errors (e.g. overcurrent or overvoltage). An interrupt/event/break generation can be invoked.

Input data into a short-circuit detector is taken from channel transceiver outputs.

There is an upcounting counter on each input channel which is counting consecutive 0's or 1's on serial data receiver outputs. A counter is restarted if there is a change in the data stream received - 1 to 0 or 0 to 1 change of data signal. If this counter reaches a short-circuit threshold register value (SCDT[7:0] bits in DFSDM\_CHyAWSCDR register), then a short-

circuit event is invoked. Each input channel has its short-circuit detector. Any channel can be selected to be continuously monitored by setting the SCDEN bit (in DFSDM\_CHyCFGR1 register) and it has its own short-circuit detector settings (threshold value in SCDT[7:0] bits, status bit SCDF[7:0], status clearing bits CLRSCDF[7:0]). Status flag SCDF[y] is cleared also by hardware when corresponding channel y is disabled (CHEN[y] = 0).

On each channel, a short-circuit detector event can be assigned to break output signal dfsdm\_break[3:0]. There are four break outputs to be assigned to a short-circuit detector event. The break signal assignment to a given channel short-circuit detector event is done by BKSCD[3:0] field in DFSDM\_CHyAWSCDR register.

Short circuit detector cannot be used in case of parallel input data channel selection (DATMPX[1:0] ≠ 0 in DFSDM\_CHyCFGR1 register).

Four break outputs are totally available (shared with the analog watchdog function).

### 27.4.12 Extreme detector

The purpose of an extremes detector is to collect the minimum and maximum values of final output data words (peak to peak values).

If the output data word is higher than the value stored in the extremes detector maximum register (EXMAX[23:0] bits in DFSDM\_FLTxEXMAX register), then this register is updated with the current output data word value and the channel from which the data is stored is in EXMAXCH[2:0] bits (in DFSDM\_FLTxEXMAX register) .

If the output data word is lower than the value stored in the extremes detector minimum register (EXMIN[23:0] bits in DFSDM\_FLTxEXMIN register), then this register is updated with the current output data word value and the channel from which the data is stored is in EXMINCH[2:0] bits (in DFSDM\_FLTxEXMIN register).

The minimum and maximum register values can be refreshed by software (by reading given DFSDM\_FLTxEXMAX or DFSDM\_FLTxEXMIN register). After refresh, the extremes detector minimum data register DFSDM\_FLTxEXMIN is filled with 0x7FFFFFFF (maximum positive value) and the extremes detector maximum register DFSDM\_FLTxEXMAX is filled with 0x800000 (minimum negative value).

The extremes detector performs a comparison after a right bit shift and an offset data correction. For each extremes detector, the input channels to be considered into computing the extremes value are selected in EXCH[7:0] bits (in DFSDM\_FLTxCR2 register).

### 27.4.13 Data unit block

The data unit block is the last block of the whole processing path: External  $\Sigma\Delta$  modulators - Serial transceivers - Sinc filter - Integrator - Data unit block.

The output data rate depends on the serial data stream rate, and filter and integrator settings. The maximum output data rate is:

$$\text{Datarate}[\text{samples} / \text{s}] = \frac{f_{\text{CKIN}}}{F_{\text{OSR}} \cdot (I_{\text{OSR}} - 1 + F_{\text{ORD}}) + (F_{\text{ORD}} + 1)} \quad \dots \text{FAST} = 0, \text{ Sincx filter}$$

$$\text{Datarate}[\text{samples} / \text{s}] = \frac{f_{\text{CKIN}}}{F_{\text{OSR}} \cdot (I_{\text{OSR}} - 1 + 4) + (2 + 1)} \quad \dots \text{FAST} = 0, \text{ FastSinc filter}$$

or

$$\text{Datarate}[\text{samples} / \text{s}] = \frac{f_{\text{CKIN}}}{F_{\text{OSR}} \cdot I_{\text{OSR}}} \quad \dots \text{FAST} = 1$$

Maximum output data rate in case of parallel data input:

$$\text{Datarate}[\text{samples} / \text{s}] = \frac{f_{\text{DATAIN\_RATE}}}{F_{\text{OSR}} \cdot (I_{\text{OSR}} - 1 + F_{\text{ORD}}) + (F_{\text{ORD}} + 1)} \quad \dots \text{FAST} = 0, \text{ Sincx filter}$$

or

$$\text{Datarate}[\text{samples} / \text{s}] = \frac{f_{\text{DATAIN\_RATE}}}{F_{\text{OSR}} \cdot (I_{\text{OSR}} - 1 + 4) + (2 + 1)} \quad \dots \text{FAST} = 0, \text{ FastSinc filter}$$

or

$$\text{Datarate}[\text{samples} / \text{s}] = \frac{f_{\text{DATAIN\_RATE}}}{F_{\text{OSR}} \cdot I_{\text{OSR}}} \quad \dots \text{FAST}=1 \text{ or any filter bypass case } (F_{\text{OSR}} = 1)$$

where:  $f_{\text{DATAIN\_RATE}}$ ...input data rate from ADC or from CPU/DMA

The right bit-shift of final data is performed in this module because the final data width is 24-bit and data coming from the processing path can be up to 32 bits. This right bit-shift is configurable in the range 0-31 bits for each selected input channel (see DTRBS[4:0] bits in DFSDM\_CHyCFGR2 register). The right bit-shift is rounding the result to nearest integer value. The sign of shifted result is maintained - to have valid 24-bit signed format of result data.

In the next step, an offset correction of the result is performed. The offset correction value (OFFSET[23:0] stored in register DFSDM\_CHyCFGR2) is subtracted from the output data for a given channel. Data in the OFFSET[23:0] field is set by software by the appropriate calibration routine.

Due to the fact that all operations in digital processing are performed on 32-bit signed registers, the following conditions must be fulfilled not to overflow the result:

$$F_{\text{OSR}}^{F_{\text{ORD}}} \cdot I_{\text{OSR}} \leq 2^{31} \quad \dots \text{ for Sinc}^x \text{ filters, } x = 1..5$$

$$2 \cdot F_{\text{OSR}}^2 \cdot I_{\text{OSR}} \leq 2^{31} \quad \dots \text{ for FastSinc filter)}$$

*Note:* In case of filter and integrator bypass ( $I_{\text{OSR}}[7:0]=0$ ,  $F_{\text{OSR}}[9:0]=0$ ), the input data rate ( $f_{\text{DATAIN\_RATE}}$ ) must be limited to be able to read all output data:

$$f_{\text{DATAIN\_RATE}} \leq f_{\text{APB}}$$

where  $f_{\text{APB}}$  is the bus frequency to which the DFSDM peripheral is connected.

#### 27.4.14 Signed data format

Each DFSDM input serial channel can be connected to one external  $\Sigma\Delta$  modulator. An external  $\Sigma\Delta$  modulator can have 2 differential inputs (positive and negative) which can be used for a differential or single-ended signal measurement.

A  $\Sigma\Delta$  modulator output is always assumed in a signed format (a data stream of zeros and ones from a  $\Sigma\Delta$  modulator represents values -1 and +1).

**Signed data format in registers:** Data is in a signed format in registers for final output data, analog watchdog, extremes detector, offset correction. The msb of output data word represents the sign of value (two's complement format).

### 27.4.15 Launching conversions

**Injected conversions** can be launched using the following methods:

- Software: writing '1' to JSWSTART in the DFSDM\_FLTxCR1 register.
- Trigger: JEXTSEL[4:0] selects the trigger signal while JEXTEN activates the trigger and selects the active edge at the same time (see the DFSDM\_FLTxCR1 register).
- Synchronous with DFSDM\_FLT0 if JSYNC=1: for DFSDM\_FLTx ( $x>0$ ), an injected conversion is automatically launched when in DFSDM\_FLT0; the injected conversion is started by software (JSWSTART=1 in DFSDM\_FLT0CR2 register). Each injected conversion in DFSDM\_FLTx ( $x>0$ ) is always executed according to its local configuration settings (JSCAN, JCHG, etc.).

If the scan conversion is enabled (bit JSCAN=1) then, each time an injected conversion is triggered, all of the selected channels in the injected group (JCHG[7:0] bits in DFSDM\_FLTxJCHGR register) are converted sequentially, starting with the lowest channel (channel 0, if selected).

If the scan conversion is disabled (bit JSCAN=0) then, each time an injected conversion is triggered, only one of the selected channels in the injected group (JCHG[7:0] bits in DFSDM\_FLTxJCHGR register) is converted and the channel selection is then moved to the next selected channel. Writing to the JCHG[7:0] bits when JSCAN=0 sets the channel selection to the lowest selected injected channel.

Only one injected conversion can be ongoing at a given time. Thus, any request to launch an injected conversion is ignored if another request for an injected conversion has already been issued but not yet completed.

**Regular conversions** can be launched using the following methods:

- Software: by writing '1' to RSWSTART in the DFSDM\_FLTxCR1 register.
- Synchronous with DFSDM\_FLT0 if RSYNC=1: for DFSDM\_FLTx ( $x>0$ ), a regular conversion is automatically launched when in DFSDM\_FLT0; a regular conversion is started by software (RSWSTART=1 in DFSDM\_FLT0CR2 register). Each regular conversion in DFSDM\_FLTx ( $x>0$ ) is always executed according to its local configuration settings (RCONT, RCH, etc.).

Only one regular conversion can be pending or ongoing at a given time. Thus, any request to launch a regular conversion is ignored if another request for a regular conversion has already been issued but not yet completed. A regular conversion can be pending if it was interrupted by an injected conversion or if it was started while an injected conversion was in progress. This pending regular conversion is then delayed and is performed when all injected conversion are finished. Any delayed regular conversion is signaled by RPEND bit in DFSDM\_FLTxRDATAR register.

### 27.4.16 Continuous and fast continuous modes

Setting RCONT in the DFSDM\_FLTxCR1 register causes regular conversions to execute in continuous mode. RCONT=1 means that the channel selected by RCH[2:0] is converted repeatedly after '1' is written to RSWSTART.

The regular conversions executing in continuous mode can be stopped by writing '0' to RCONT. After clearing RCONT, the on-going conversion is stopped immediately.

In continuous mode, the data rate can be increased by setting the FAST bit in the DFSDM\_FLTxCR1 register. In this case, the filter does not need to be refilled by new fresh data if converting continuously from one channel because data inside the filter is valid from previously sampled continuous data. The speed increase depends on the chosen filter order. The first conversion in fast mode (FAST=1) after starting a continuous conversion by RSWSTART=1 takes still full time (as when FAST=0), then each subsequent conversion is finished in shorter intervals.

Conversion time in continuous mode:

if FAST = 0 (or first conversion if FAST=1):

for Sinc<sup>X</sup> filters:

$$t = \text{CNVCNT} / f_{\text{DFSDMCLK}} = [F_{\text{OSR}} * (I_{\text{OSR}} - 1 + F_{\text{ORD}}) + F_{\text{ORD}}] / f_{\text{CKIN}}$$

for FastSinc filter:

$$t = \text{CNVCNT} / f_{\text{DFSDMCLK}} = [F_{\text{OSR}} * (I_{\text{OSR}} - 1 + 4) + 2] / f_{\text{CKIN}}$$

if FAST = 1 (except first conversion):

for Sinc<sup>X</sup> and FastSinc filters:

$$t = \text{CNVCNT} / f_{\text{DFSDMCLK}} = [F_{\text{OSR}} * I_{\text{OSR}}] / f_{\text{CKIN}}$$

in case  $F_{\text{OSR}} = \text{FOSR}[9:0] + 1 = 1$  (filter bypassed, only integrator active):

$$t = I_{\text{OSR}} / f_{\text{CKIN}} \text{ (... but CNVCNT=0)}$$

Continuous mode is not available for injected conversions. Injected conversions can be started by timer trigger to emulate the continuous mode with precise timing.

If a regular continuous conversion is in progress (RCONT=1) and if a write access to DFSDM\_FLTxCR1 register requesting regular continuous conversion (RCONT=1) is performed, then regular continuous conversion is restarted from the next conversion cycle (like new regular continuous conversion is applied for new channel selection - even if there is no change in DFSDM\_FLTxCR1 register).

### 27.4.17 Request precedence

An injected conversion has a higher precedence than a regular conversion. A regular conversion which is already in progress is immediately interrupted by the request of an injected conversion; this regular conversion is restarted after the injected conversion finishes.

An injected conversion cannot be launched if another injected conversion is pending or already in progress: any request to launch an injected conversion (either by JSWSTART or by a trigger) is ignored as long as bit JCIP is '1' (in the DFSDM\_FLTxISR register).

Similarly, a regular conversion cannot be launched if another regular conversion is pending or already in progress: any request to launch a regular conversion (using RSWSTART) is ignored as long as bit RCIP is '1' (in the DFSDM\_FLTxISR register).

However, if an injected conversion is requested while a regular conversion is already in progress, the regular conversion is immediately stopped and an injected conversion is launched. The regular conversion is then restarted and this delayed restart is signaled in bit RPEND.

Injected conversions have precedence over regular conversions in that a injected conversion can temporarily interrupt a sequence of continuous regular conversions. When

the sequence of injected conversions finishes, the continuous regular conversions start again if RCONT is still set (and RPEND bit will signalize the delayed start on the first regular conversion result).

Precedence also matters when actions are initiated by the same write to DFSDM, or if multiple actions are pending at the end of another action. For example, suppose that, while an injected conversion is in process (JCIP=1), a single write operation to DFSDM\_FLTxCR1 writes '1' to RSWSTART, requesting a regular conversion. When the injected sequence finishes, the precedence dictates that the regular conversion is performed next and its delayed start is signalized in RPEND bit.

### 27.4.18 Power optimization in run mode

In order to reduce the consumption, the DFSDM filter and integrator are automatically put into idle when not used by conversions (RCIP=0, JCIP=0).

## 27.5 DFSDM interrupts

In order to increase the CPU performance, a set of interrupts related to the CPU event occurrence has been implemented:

- End of injected conversion interrupt:
  - enabled by JEOCIE bit in DFSDM\_FLTxCR2 register
  - indicated in JEOCF bit in DFSDM\_FLTxISR register
  - cleared by reading DFSDM\_FLTxJDATAR register (injected data)
  - indication of which channel end of conversion occurred, reported in JDATACH[2:0] bits in DFSDM\_FLTxJDATAR register
- End of regular conversion interrupt:
  - enabled by REOCIE bit in DFSDM\_FLTxCR2 register
  - indicated in REOCF bit in DFSDM\_FLTxISR register
  - cleared by reading DFSDM\_FLTxRDATAR register (regular data)
  - indication of which channel end of conversion occurred, reported in RDATACH[2:0] bits in DFSDM\_FLTxRDATAR register
- Data overrun interrupt for injected conversions:
  - occurred when injected converted data were not read from DFSDM\_FLTxJDATAR register (by CPU or DMA) and were overwritten by a new injected conversion
  - enabled by JOVRIE bit in DFSDM\_FLTxCR2 register
  - indicated in JOVRF bit in DFSDM\_FLTxISR register
  - cleared by writing '1' into CLRJOVRF bit in DFSDM\_FLTxICR register
- Data overrun interrupt for regular conversions:
  - occurred when regular converted data were not read from DFSDM\_FLTxRDATAR register (by CPU or DMA) and were overwritten by a new regular conversion
  - enabled by ROVRIE bit in DFSDM\_FLTxCR2 register
  - indicated in ROVRF bit in DFSDM\_FLTxISR register
  - cleared by writing '1' into CLRROVRF bit in DFSDM\_FLTxICR register
- Analog watchdog interrupt:

- occurred when converted data (output data or data from analog watchdog filter - according to AWFSEL bit setting in DFSDM\_FLTxCR1 register) crosses over/under high/low thresholds in DFSDM\_FLTxAWHTR / DFSDM\_FLTxAWLTR registers
- enabled by AWDIE bit in DFSDM\_FLTxCR2 register (on selected channels AWDCH[7:0])
- indicated in AWDF bit in DFSDM\_FLTxISR register
- separate indication of high or low analog watchdog threshold error by AWHTF[7:0] and AWLTF[7:0] fields in DFSDM\_FLTxAWSR register
- cleared by writing '1' into corresponding CLRAWHTF[7:0] or CLRAWLTF[7:0] bits in DFSDM\_FLTxAWCFR register
- Short-circuit detector interrupt:
  - occurred when the number of stable data crosses over thresholds in DFSDM\_CHyAWSCDR register
  - enabled by SCDIE bit in DFSDM\_FLTxCR2 register (on channel selected by SCDEN bit in DFSDM\_CHyCFGR1 register)
  - indicated in SCDF[7:0] bits in DFSDM\_FLTxISR register (which also reports the channel on which the short-circuit detector event occurred)
  - cleared by writing '1' into the corresponding CLRSCDF[7:0] bit in DFSDM\_FLTxICR register
- Channel clock absence interrupt:
  - occurred when there is clock absence on CKINy pin (see [Clock absence detection](#) in [Section 27.4.4: Serial channel transceivers](#))
  - enabled by CKABIE bit in DFSDM\_FLTxCR2 register (on channels selected by CKABEN bit in DFSDM\_CHyCFGR1 register)
  - indicated in CKABF[y] bit in DFSDM\_FLTxISR register
  - cleared by writing '1' into CLRCKABF[y] bit in DFSDM\_FLTxICR register

**Table 168. DFSDM interrupt requests**

Interrupt event	Event flag	Event/Interrupt clearing method	Interrupt enable control bit
End of injected conversion	JEOCF	reading DFSDM_FLTxJDATAR	JEOCIE
End of regular conversion	REOCF	reading DFSDM_FLTxRDATAR	REOCIE
Injected data overrun	JOVRF	writing CLRJOVRF = 1	JOVRIE
Regular data overrun	ROVRF	writing CLRROVRF = 1	ROVRIE
Analog watchdog	AWDF, AWHTF[7:0], AWLTF[7:0]	writing CLRAWHTF[7:0] = 1 writing CLRAWLTF[7:0] = 1	AWDIE, (AWDCH[7:0])
short-circuit detector	SCDF[7:0]	writing CLRSCDF[7:0] = 1	SCDIE, (SCDEN)
Channel clock absence	CKABF[7:0]	writing CLRCKABF[7:0] = 1	CKABIE, (CKABEN)



## 27.6 DFSDM DMA transfer

To decrease the CPU intervention, conversions can be transferred into memory using a DMA transfer. A DMA transfer for injected conversions is enabled by setting bit JDMAEN=1 in DFSDM\_FLTxCR1 register. A DMA transfer for regular conversions is enabled by setting bit RDMAEN=1 in DFSDM\_FLTxCR1 register.

*Note:* With a DMA transfer, the interrupt flag is automatically cleared at the end of the injected or regular conversion (JEOCF or REOCF bit in DFSDM\_FLTxISR register) because DMA is reading DFSDM\_FLTxJDATAR or DFSDM\_FLTxRDATAR register.

## 27.7 DFSDM channel y registers (y=0..7)

### 27.7.1 DFSDM channel y configuration register (DFSDM\_CHyCFGR1)

This register specifies the parameters used by channel y.

Address offset: 0x00 + 0x20 \* y, (y = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DFSDM EN	CKOUT SRC	Res.	Res.	Res.	Res.	Res.	Res.	CKOUTDIV[7:0]							
rw	rw							rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHIN SEL	CHEN	CKAB EN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
rw	rw	rw	rw				rw	rw	rw	rw		rw	rw	rw	rw

Bit 31 **DFSDMEN**: Global enable for DFSDM interface

- 0: DFSDM interface disabled
- 1: DFSDM interface enabled

If DFSDM interface is enabled, then it is started to operate according to enabled y channels and enabled x filters settings (CHEN bit in DFSDM\_CHyCFGR1 and DFEN bit in DFSDM\_FLTxCR1).

Data cleared by setting DFSDMEN=0:

- all registers DFSDM\_FLTxISR are set to reset state (x = 0..3)
- all registers DFSDM\_FLTxAWSR are set to reset state (x = 0..3)

*Note:* DFSDMEN is present only in DFSDM\_CH0CFGR1 register (channel y=0)

Bit 30 **CKOUTSRC**: Output serial clock source selection

- 0: Source for output clock is from system clock
- 1: Source for output clock is from audio clock

-SAI1 clock selected by SAI1SEL[1:0] field in RCC configuration (see [Section 6.4.32: Peripherals independent clock configuration register \(RCC\\_CCIPR2\)](#))

This value can be modified only when DFSDMEN=0 (in DFSDM\_CH0CFGR1 register).

*Note:* CKOUTSRC is present only in DFSDM\_CH0CFGR1 register (channel y=0)

Bits 29:24 Reserved, must be kept at reset value.



Bits 23:16 **CKOUTDIV[7:0]**: Output serial clock divider

0: Output clock generation is disabled (CKOUT signal is set to low state)

1- 255: Defines the division of system clock for the serial clock output for CKOUT signal in range 2 - 256 (Divider = CKOUTDIV+1).

CKOUTDIV also defines the threshold for a clock absence detection.

This value can only be modified when DFSDMEN=0 (in DFSDM\_CH0CFGR1 register).

If DFSDMEN=0 (in DFSDM\_CH0CFGR1 register) then CKOUT signal is set to low state (setting is performed one DFSDM clock cycle after DFSDMEN=0).

*Note: CKOUTDIV is present only in DFSDM\_CH0CFGR1 register (channel y=0)*

Bits 15:14 **DATPACK[1:0]**: Data packing mode in DFSDM\_CHyDATINR register.

0: Standard: input data in DFSDM\_CHyDATINR register are stored only in INDAT0[15:0]. To empty DFSDM\_CHyDATINR register one sample must be read by the DFSDM filter from channel y.

1: Interleaved: input data in DFSDM\_CHyDATINR register are stored as two samples:

–first sample in INDAT0[15:0] (assigned to channel y)

–second sample INDAT1[15:0] (assigned to channel y)

To empty DFSDM\_CHyDATINR register, two samples must be read by the digital filter from channel y (INDAT0[15:0] part is read as first sample and then INDAT1[15:0] part is read as next sample).

2: Dual: input data in DFSDM\_CHyDATINR register are stored as two samples:

–first sample INDAT0[15:0] (assigned to channel y)

–second sample INDAT1[15:0] (assigned to channel y+1)

To empty DFSDM\_CHyDATINR register first sample must be read by the digital filter from channel y and second sample must be read by another digital filter from channel y+1. Dual mode is available only on even channel numbers (y = 0, 2, 4, 6), for odd channel numbers (y = 1, 3, 5, 7) DFSDM\_CHyDATINR is write protected. If an even channel is set to dual mode then the following odd channel must be set into standard mode (DATPACK[1:0]=0) for correct cooperation with even channel.

3: Reserved

This value can be modified only when CHEN=0 (in DFSDM\_CHyCFGR1 register).

Bits 13:12 **DATMPX[1:0]**: Input data multiplexer for channel y

0: Data to channel y are taken from external serial inputs as 1-bit values. DFSDM\_CHyDATINR register is write protected.

1: Data to channel y are taken from internal analog to digital converter ADC<sub>y+1</sub> output register update as 16-bit values (if ADC<sub>y+1</sub> is available). Data from ADCs are written into INDAT0[15:0] part of DFSDM\_CHyDATINR register.

2: Data to channel y are taken from internal DFSDM\_CHyDATINR register by direct CPU/DMA write. There can be written one or two 16-bit data samples according DATPACK[1:0] bit field setting.

3: Reserved

*Note: This value can be modified only when CHEN=0 (in DFSDM\_CHyCFGR1 register).*

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

Bit 8 **CHINSEL**: Channel inputs selection

0: Channel inputs are taken from pins of the same channel y.

1: Channel inputs are taken from pins of the following channel (channel (y+1) modulo 8).

This value can be modified only when CHEN=0 (in DFSDM\_CHyCFGR1 register).

Bit 7 **CHEN**: Channel y enable

0: Channel y disabled

1: Channel y enabled

If channel y is enabled, then serial data receiving is started according to the given channel setting.

Bit 6 **CKABEN**: Clock absence detector enable on channel y

- 0: Clock absence detector disabled on channel y
- 1: Clock absence detector enabled on channel y

Bit 5 **SCDEN**: Short-circuit detector enable on channel y

- 0: Input channel y will not be guarded by the short-circuit detector
- 1: Input channel y will be continuously guarded by the short-circuit detector

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **SPICKSEL[1:0]**: SPI clock select for channel y

- 0: clock coming from external CKINy input - sampling point according SITP[1:0]
- 1: clock coming from internal CKOUT output - sampling point according SITP[1:0]
- 2: clock coming from internal CKOUT - sampling point on each second CKOUT falling edge.  
For connection to external  $\Sigma\Delta$  modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input rising edge).
- 3: clock coming from internal CKOUT output - sampling point on each second CKOUT rising edge.  
For connection to external  $\Sigma\Delta$  modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input falling edge).

This value can be modified only when CHEN=0 (in DFSDM\_CHyCFGR1 register).

Bits 1:0 **SITP[1:0]**: Serial interface type for channel y

- 00: SPI with rising edge to strobe data
  - 01: SPI with falling edge to strobe data
  - 10: Manchester coded input on DATINy pin: rising edge = logic 0, falling edge = logic 1
  - 11: Manchester coded input on DATINy pin: rising edge = logic 1, falling edge = logic 0
- This value can only be modified when CHEN=0 (in DFSDM\_CHyCFGR1 register).

### 27.7.2 DFSDM channel y configuration register (DFSDM\_CHyCFGR2)

This register specifies the parameters used by channel y.

Address offset:  $0x04 + 0x20 * y$ , (y = 0 to 7)

Reset value: 0x0000 0000

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

Bits 31:8 **OFFSET[23:0]**: 24-bit calibration offset for channel y

For channel y, OFFSET is applied to the results of each conversion from this channel.  
This value is set by software.

Bits 7:3 **DTRBS[4:0]**: Data right bit-shift for channel y

0-31: Defines the shift of the data result coming from the integrator - how many bit shifts to the right will be performed to have final results. Bit-shift is performed before offset correction. The data shift is rounding the result to nearest integer value. The sign of shifted result is maintained (to have valid 24-bit signed format of result data).

This value can be modified only when CHEN=0 (in DFSDM\_CHyCFGR1 register).

Bits 2:0 Reserved, must be kept at reset value.

### 27.7.3 DFSDM channel y analog watchdog and short-circuit detector register (DFSDM\_CHyAWSCDR)

Short-circuit detector and analog watchdog settings for channel y.

Address offset: 0x08 + 0x20 \* y, (y = 0 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.	AWFORD[1:0]		Res.	AWFOSR[4:0]				
								r/w	r/w		r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
r/w	r/w	r/w	r/w					r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 23:22 **AWFORD[1:0]**: Analog watchdog Sinc filter order on channel y

0: FastSinc filter type

1: Sinc<sup>1</sup> filter type

2: Sinc<sup>2</sup> filter type

3: Sinc<sup>3</sup> filter type

Sinc<sup>x</sup> filter type transfer function:

$$H(z) = \left( \frac{1 - z^{-FOSR}}{1 - z^{-1}} \right)^x$$

FastSinc filter type transfer function:

$$H(z) = \left( \frac{1 - z^{-FOSR}}{1 - z^{-1}} \right)^2 \cdot (1 + z^{-(2 \cdot FOSR)})$$

This bit can be modified only when CHEN=0 (in DFSDM\_CHyCFGR1 register).

Bit 21 Reserved, must be kept at reset value.

Bits 20:16 **AWFOSR[4:0]**: Analog watchdog filter oversampling ratio (decimation rate) on channel y

0 - 31: Defines the length of the Sinc type filter in the range 1 - 32 (AWFOSR + 1). This number is also the decimation ratio of the analog data rate.

This bit can be modified only when CHEN=0 (in DFSDM\_CHyCFGR1 register).

Note: If AWFOSR = 0 then the filter has no effect (filter bypass).

Bits 15:12 **BKSCD[3:0]**: Break signal assignment for short-circuit detector on channel y  
 BKSCD[i] = 0: Break i signal not assigned to short-circuit detector on channel y  
 BKSCD[i] = 1: Break i signal assigned to short-circuit detector on channel y

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:0 **SCDT[7:0]**: short-circuit detector threshold for channel y  
 These bits are written by software to define the threshold counter for the short-circuit detector. If this value is reached, then a short-circuit detector event occurs on a given channel.

### 27.7.4 DFSDM channel y watchdog filter data register (DFSDM\_CHyWDATR)

This register contains the data resulting from the analog watchdog filter associated to the input channel y.

Address offset: 0x0C + 0x20 \* y, (y = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDATA[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 **WDATA[15:0]**: Input channel y watchdog data  
 Data converted by the analog watchdog filter for input channel y. This data is continuously converted (no trigger) for this channel, with a limited resolution (OSR=1..32/sinc order = 1..3).

### 27.7.5 DFSDM channel y data input register (DFSDM\_CHyDATINR)

This register contains 16-bit input data to be processed by DFSDM filter module.

Address offset: 0x10 + 0x20 \* y, (y = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INDAT1[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
INDAT0[15:0]															
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Bits 31:16 **INDAT1[15:0]**: Input data for channel y or channel y+1

Input parallel channel data to be processed by the digital filter if DATMPX[1:0]=1 or DATMPX[1:0]=2. Data can be written by CPU/DMA (if DATMPX[1:0]=2) or directly by internal ADC (if DATMPX[1:0]=1).

If DATPACK[1:0]=0 (standard mode)

INDAT0[15:0] is write protected (not used for input sample).

If DATPACK[1:0]=1 (interleaved mode)

Second channel y data sample is stored into INDAT1[15:0]. First channel y data sample is stored into INDAT0[15:0]. Both samples are read sequentially by DFSDM\_FLTx filter as two channel y data samples.

If DATPACK[1:0]=2 (dual mode).

For even y channels: sample in INDAT1[15:0] is automatically copied into INDAT0[15:0] of channel (y+1).

For odd y channels: INDAT1[15:0] is write protected.

See [Section 27.4.6: Parallel data inputs](#) for more details.

INDAT0[15:1] is in the 16-bit signed format.

Bits 15:0 **INDAT0[15:0]**: Input data for channel y

Input parallel channel data to be processed by the digital filter if DATMPX[1:0]=1 or DATMPX[1:0]=2. Data can be written by CPU/DMA (if DATMPX[1:0]=2) or directly by internal ADC (if DATMPX[1:0]=1).

If DATPACK[1:0]=0 (standard mode)

Channel y data sample is stored into INDAT0[15:0].

If DATPACK[1:0]=1 (interleaved mode)

First channel y data sample is stored into INDAT0[15:0]. Second channel y data sample is stored into INDAT1[15:0]. Both samples are read sequentially by DFSDM\_FLTx filter as two channel y data samples.

If DATPACK[1:0]=2 (dual mode).

For even y channels: Channel y data sample is stored into INDAT0[15:0].

For odd y channels: INDAT0[15:0] is write protected.

See [Section 27.4.6: Parallel data inputs](#) for more details.

INDAT0[15:0] is in the 16-bit signed format.

### 27.7.6 DFSDM channel y delay register (DFSDM\_CHyDLYR)

Address offset: 0x14 + 0x20 \* y, (y = 0 to 7)

Reset value: 0x0000 0000

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

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

Bits 5:0 **PLSSKP[5:0]**: Pulses to skip for input data skipping function

0-63: Defines the number of serial input samples that will be skipped. Skipping is applied immediately after writing to this field. Reading of PLSSKP[5:0] returns current value of pulses which will be skipped. If PLSSKP[5:0]=0 then all required data samples were already skipped.

Note: User can update PLSSKP[5:0] also when PLSSKP[5:0] is not zero.

## 27.8 DFSDM filter x module registers (x=0..3)

### 27.8.1 DFSDM filter x control register 1 (DFSDM\_FLTxCR1)

Address offset: 0x100 + 0x80 \* 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.	AWFSEL	FAST	Res.	Res.	RCH[2:0]			Res.	Res.	RDMAEN	Res.	RSYNC	RCON T	RSW START	Res.
	rw	rw			rw	rw	rw			rw		rw	rw	rt_w1	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	JEXTEN[1:0]		JEXTSEL[4:0]				Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSW START	DFEN	
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw		rt_w1	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **AWFSEL**: Analog watchdog fast mode select

0: Analog watchdog on data output value (after the digital filter). The comparison is done after offset correction and shift

1: Analog watchdog on channel transceivers value (after watchdog filter)

Bit 29 **FAST**: Fast conversion mode selection for regular conversions

0: Fast conversion mode disabled

1: Fast conversion mode enabled

When converting a regular conversion in continuous mode, having enabled the fast mode causes each conversion (except the first) to execute faster than in standard mode. This bit has no effect on conversions which are not continuous.

This bit can be modified only when DFEN=0 (DFSDM\_FLTxCR1).

if FAST=0 (or first conversion in continuous mode if FAST=1):

$$t = [F_{OSR} * (I_{OSR}-1 + F_{ORD}) + F_{ORD}] / f_{CKIN} \dots \text{ for Sinc}^x \text{ filters}$$

$$t = [F_{OSR} * (I_{OSR}-1 + 4) + 2] / f_{CKIN} \dots \text{ for FastSinc filter}$$

if FAST=1 in continuous mode (except first conversion):

$$t = [F_{OSR} * I_{OSR}] / f_{CKIN}$$

in case if  $F_{OSR} = F_{OSR}[9:0]+1 = 1$  (filter bypassed, active only integrator):

$$t = I_{OSR} / f_{CKIN} \dots \text{ but CNVCNT}=0$$

where:  $f_{CKIN}$  is the channel input clock frequency (on given channel CKINy pin) or input data rate in case of parallel data input.

Bits 28:27 Reserved, must be kept at reset value.

Bits 26:24 **RCH[2:0]**: Regular channel selection

0: Channel 0 is selected as the regular channel

1: Channel 1 is selected as the regular channel

...

7: Channel 7 is selected as the regular channel

Writing these bits when RCIP=1 takes effect when the next regular conversion begins. This is especially useful in continuous mode (when RCONT=1). It also affects regular conversions which are pending (due to ongoing injected conversion).

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **RDMAEN**: DMA channel enabled to read data for the regular conversion

0: The DMA channel is not enabled to read regular data

1: The DMA channel is enabled to read regular data

This bit can be modified only when DFEN=0 (DFSDM\_FLTxCR1).

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RSYNC**: Launch regular conversion synchronously with DFSDM\_FLT0

0: Do not launch a regular conversion synchronously with DFSDM\_FLT0

1: Launch a regular conversion in this DFSDM\_FLTx at the very moment when a regular conversion is launched in DFSDM\_FLT0

This bit can be modified only when DFEN=0 (DFSDM\_FLTxCR1).

Bit 18 **RCONT**: Continuous mode selection for regular conversions

0: The regular channel is converted just once for each conversion request

1: The regular channel is converted repeatedly after each conversion request

Writing '0' to this bit while a continuous regular conversion is already in progress stops the continuous mode immediately.

Bit 17 **RSWSTART**: Software start of a conversion on the regular channel

0: Writing '0' has no effect

1: Writing '1' makes a request to start a conversion on the regular channel and causes RCIP to become '1'. If RCIP=1 already, writing to RSWSTART has no effect. Writing '1' has no effect if RSYNC=1.

This bit is always read as '0'.

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:13 **JEXTEN[1:0]**: Trigger enable and trigger edge selection for injected conversions

00: Trigger detection is disabled

01: Each rising edge on the selected trigger makes a request to launch an injected conversion

10: Each falling edge on the selected trigger makes a request to launch an injected conversion

11: Both rising edges and falling edges on the selected trigger make requests to launch injected conversions

This bit can be modified only when DFEN=0 (DFSDM\_FLTxCR1).

Bits 12:8 **JEXTSEL[4:0]**: Trigger signal selection for launching injected conversions

0x0-0x1F: Trigger inputs selected by the following table (internal or external trigger).

This bit can be modified only when DFEN=0 (DFSDM\_FLTxCR1).

*Note: synchronous trigger has latency up to one  $f_{DFSDMCLK}$  clock cycle (with deterministic jitter), asynchronous trigger has latency 2-3  $f_{DFSDMCLK}$  clock cycles (with jitter up to 1 cycle).*

	DFSDM_FLT0	DFSDM_FLT1	DFSDM_FLT2	DFSDM_FLT3
0x00	dfsdm_jtrg0	dfsdm_jtrg0	dfsdm_jtrg0	dfsdm_jtrg0
0x01	dfsdm_jtrg1	dfsdm_jtrg1	dfsdm_jtrg1	dfsdm_jtrg1
...				
0x1E	dfsdm_jtrg30	dfsdm_jtrg30	dfsdm_jtrg30	dfsdm_jtrg30
0x1F	dfsdm_jtrg31	dfsdm_jtrg31	dfsdm_jtrg31	dfsdm_jtrg31

Refer to [Table 164: DFSDM triggers connection](#).

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

Bit 5 **JDMAEN**: DMA channel enabled to read data for the injected channel group

0: The DMA channel is not enabled to read injected data

1: The DMA channel is enabled to read injected data

This bit can be modified only when DFEN=0 (DFSDM\_FLTxCR1).

Bit 4 **JSCAN**: Scanning conversion mode for injected conversions

0: One channel conversion is performed from the injected channel group and next the selected channel from this group is selected.

1: The series of conversions for the injected group channels is executed, starting over with the lowest selected channel.

This bit can be modified only when DFEN=0 (DFSDM\_FLTxCR1).

Writing JCHG if JSCAN=0 resets the channel selection to the lowest selected channel.

Bit 3 **JSYNC**: Launch an injected conversion synchronously with the DFSDM\_FLT0 JSWSTART trigger

0: Do not launch an injected conversion synchronously with DFSDM\_FLT0

1: Launch an injected conversion in this DFSDM\_FLTx at the very moment when an injected conversion is launched in DFSDM\_FLT0 by its JSWSTART trigger

This bit can be modified only when DFEN=0 (DFSDM\_FLTxCR1).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **JSWSTART**: Start a conversion of the injected group of channels

0: Writing '0' has no effect.

1: Writing '1' makes a request to convert the channels in the injected conversion group, causing JCIP to become '1' at the same time. If JCIP=1 already, then writing to JSWSTART has no effect. Writing '1' has no effect if JSYNC=1.

This bit is always read as '0'.

Bit 0 **DFEN**: DFSDM\_FLTx enable

0: DFSDM\_FLTx is disabled. All conversions of given DFSDM\_FLTx are stopped immediately and all DFSDM\_FLTx functions are stopped.

1: DFSDM\_FLTx is enabled. If DFSDM\_FLTx is enabled, then DFSDM\_FLTx starts operating according to its setting.

Data which are cleared by setting DFEN=0:

–register DFSDM\_FLTxISR is set to the reset state

–register DFSDM\_FLTxAWSR is set to the reset state



**27.8.2 DFSDM filter x control register 2 (DFSDM\_FLTxCR2)**

Address offset: 0x104 + 0x80 \* x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[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
EXCH[7:0]								Res.	CKABIE	SCDIE	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
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 23:16 **AWDCH[7:0]**: Analog watchdog channel selection

These bits select the input channel to be guarded continuously by the analog watchdog.

AWDCH[y] = 0: Analog watchdog is disabled on channel y

AWDCH[y] = 1: Analog watchdog is enabled on channel y

Bits 15:8 **EXCH[7:0]**: Extremes detector channel selection

These bits select the input channels to be taken by the Extremes detector.

EXCH[y] = 0: Extremes detector does not accept data from channel y

EXCH[y] = 1: Extremes detector accepts data from channel y

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CKABIE**: Clock absence interrupt enable

0: Detection of channel input clock absence interrupt is disabled

1: Detection of channel input clock absence interrupt is enabled

Please see the explanation of CKABF[7:0] in DFSDM\_FLTxISR.

*Note: CKABIE is present only in DFSDM\_FLT0CR2 register (filter x=0)*

Bit 5 **SCDIE**: Short-circuit detector interrupt enable

0: short-circuit detector interrupt is disabled

1: short-circuit detector interrupt is enabled

Please see the explanation of SCDF[7:0] in DFSDM\_FLTxISR.

*Note: SCDIE is present only in DFSDM\_FLT0CR2 register (filter x=0)*

Bit 4 **AWDIE**: Analog watchdog interrupt enable

0: Analog watchdog interrupt is disabled

1: Analog watchdog interrupt is enabled

Please see the explanation of AWDF in DFSDM\_FLTxISR.

Bit 3 **ROVRIE**: Regular data overrun interrupt enable

0: Regular data overrun interrupt is disabled

1: Regular data overrun interrupt is enabled

Please see the explanation of ROVRF in DFSDM\_FLTxISR.

- Bit 2 **JOVRIE**: Injected data overrun interrupt enable
  - 0: Injected data overrun interrupt is disabled
  - 1: Injected data overrun interrupt is enabled
 Please see the explanation of JOVRF in DFSDM\_FLTxISR.
- Bit 1 **REOCIE**: Regular end of conversion interrupt enable
  - 0: Regular end of conversion interrupt is disabled
  - 1: Regular end of conversion interrupt is enabled
 Please see the explanation of REOCF in DFSDM\_FLTxISR.
- Bit 0 **JEOCIE**: Injected end of conversion interrupt enable
  - 0: Injected end of conversion interrupt is disabled
  - 1: Injected end of conversion interrupt is enabled
 Please see the explanation of JEOCF in DFSDM\_FLTxISR.

### 27.8.3 DFSDM filter x interrupt and status register (DFSDM\_FLTxISR)

Address offset: 0x108 + 0x80 \* x, (x = 0 to 3)

Reset value: 0x00FF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCDF[7:0]								CKABF[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	r	r									r	r	r	r	r

Bits 31:24 **SCDF[7:0]**: short-circuit detector flag  
 SDCF[y]=0: No short-circuit detector event occurred on channel y  
 SDCF[y]=1: The short-circuit detector counter reaches, on channel y, the value programmed in the DFSDM\_CHyAWSCDR registers  
 This bit is set by hardware. It can be cleared by software using the corresponding CLRSCDF[y] bit in the DFSDM\_FLTxICR register. SCDF[y] is cleared also by hardware when CHEN[y] = 0 (given channel is disabled).  
*Note: SCDF[7:0] is present only in DFSDM\_FLT0ISR register (filter x=0)*

Bits 23:16 **CKABF[7:0]**: Clock absence flag  
 CKABF[y]=0: Clock signal on channel y is present.  
 CKABF[y]=1: Clock signal on channel y is not present.  
 Given y bit is set by hardware when clock absence is detected on channel y. It is held at CKABF[y]=1 state by hardware when CHEN=0 (see DFSDM\_CHyCFGR1 register). It is held at CKABF[y]=1 state by hardware when the transceiver is not yet synchronized. It can be cleared by software using the corresponding CLRCKABF[y] bit in the DFSDM\_FLTxICR register.  
*Note: CKABF[7:0] is present only in DFSDM\_FLT0ISR register (filter x=0)*

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RCIP**: Regular conversion in progress status  
 0: No request to convert the regular channel has been issued  
 1: The conversion of the regular channel is in progress or a request for a regular conversion is pending  
 A request to start a regular conversion is ignored when RCIP=1.



Bit 13 **JCIP**: Injected conversion in progress status

0: No request to convert the injected channel group (neither by software nor by trigger) has been issued

1: The conversion of the injected channel group is in progress or a request for a injected conversion is pending, due either to '1' being written to JSWSTART or to a trigger detection

A request to start an injected conversion is ignored when JCIP=1.

Bits 12:5 Reserved, must be kept at reset value.

Bit 4 **AWDF**: Analog watchdog

0: No Analog watchdog event occurred

1: The analog watchdog block detected voltage which crosses the value programmed in the DFSDM\_FLTxAWLTR or DFSDM\_FLTxAWHTR registers.

This bit is set by hardware. It is cleared by software by clearing all source flag bits AWHTF[7:0] and AWLTF[7:0] in DFSDM\_FLTxAWSR register (by writing '1' into the clear bits in DFSDM\_FLTxAWCFR register).

Bit 3 **ROVRF**: Regular conversion overrun flag

0: No regular conversion overrun has occurred

1: A regular conversion overrun has occurred, which means that a regular conversion finished while REOCF was already '1'. RDATAR is not affected by overruns

This bit is set by hardware. It can be cleared by software using the CLRROVRF bit in the DFSDM\_FLTxICR register.

Bit 2 **JOVRF**: Injected conversion overrun flag

0: No injected conversion overrun has occurred

1: An injected conversion overrun has occurred, which means that an injected conversion finished while JEOCF was already '1'. JDATAR is not affected by overruns

This bit is set by hardware. It can be cleared by software using the CLRJOVRF bit in the DFSDM\_FLTxICR register.

Bit 1 **REOCF**: End of regular conversion flag

0: No regular conversion has completed

1: A regular conversion has completed and its data may be read

This bit is set by hardware. It is cleared when the software or DMA reads DFSDM\_FLTxRDATAR.

Bit 0 **JEOCF**: End of injected conversion flag

0: No injected conversion has completed

1: An injected conversion has completed and its data may be read

This bit is set by hardware. It is cleared when the software or DMA reads DFSDM\_FLTxJDATAR.

*Note:* For each of the flag bits, an interrupt can be enabled by setting the corresponding bit in DFSDM\_FLTxCR2. If an interrupt is called, the flag must be cleared before exiting the interrupt service routine.

All the bits of DFSDM\_FLTxISR are automatically reset when DFEN=0.

### 27.8.4 DFSDM filter x interrupt flag clear register (DFSDM\_FLTxICR)

Address offset: 0x10C + 0x80 \* 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
CLRSCDF[7:0]								CLRCKABF[7:0]							
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
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.	CLRR OVRF	CLRJ OVRF	Res.	Res.
												rc_w1	rc_w1		

Bits 31:24 **CLRSCDF[7:0]**: Clear the short-circuit detector flag

CLRSCDF[y]=0: Writing '0' has no effect

CLRSCDF[y]=1: Writing '1' to position y clears the corresponding SCDF[y] bit in the DFSDM\_FLTxISR register

Note: CLRSCDF[7:0] is present only in DFSDM\_FLT0ICR register (filter x=0)

Bits 23:16 **CLRCKABF[7:0]**: Clear the clock absence flag

CLRCKABF[y]=0: Writing '0' has no effect

CLRCKABF[y]=1: Writing '1' to position y clears the corresponding CKABF[y] bit in the DFSDM\_FLTxISR register. When the transceiver is not yet synchronized, the clock absence flag is set and cannot be cleared by CLRCKABF[y].

Note: CLRCKABF[7:0] is present only in DFSDM\_FLT0ICR register (filter x=0)

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

Bit 3 **CLRROVRF**: Clear the regular conversion overrun flag

0: Writing '0' has no effect

1: Writing '1' clears the ROVRF bit in the DFSDM\_FLTxISR register

Bit 2 **CLRJOVRF**: Clear the injected conversion overrun flag

0: Writing '0' has no effect

1: Writing '1' clears the JOVRF bit in the DFSDM\_FLTxISR register

Bits 1:0 Reserved, must be kept at reset value.

Note: The bits of DFSDM\_FLTxICR are always read as '0'.

### 27.8.5 DFSDM filter x injected channel group selection register (DFSDM\_FLTxJCHGR)

Address offset: 0x110 + 0x80 \* x, (x = 0 to 3)

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.	Res.	Res.	Res.	Res.	JCHG[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **JCHG[7:0]**: Injected channel group selection

JCHG[y]=0: channel y is not part of the injected group

JCHG[y]=1: channel y is part of the injected group

If JSCAN=1, each of the selected channels is converted, one after another. The lowest channel (channel 0, if selected) is converted first and the sequence ends at the highest selected channel.

If JSCAN=0, then only one channel is converted from the selected channels, and the channel selection is moved to the next channel. Writing JCHG, if JSCAN=0, resets the channel selection to the lowest selected channel.

At least one channel must always be selected for the injected group. Writes causing all JCHG bits to be zero are ignored.

### 27.8.6 DFSDM filter x control register (DFSDM\_FLTxFCR)

Address offset: 0x114 + 0x80 \* 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
FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]									
rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **FORD[2:0]**: Sinc filter order

- 0: FastSinc filter type
- 1: Sinc<sup>1</sup> filter type
- 2: Sinc<sup>2</sup> filter type
- 3: Sinc<sup>3</sup> filter type
- 4: Sinc<sup>4</sup> filter type
- 5: Sinc<sup>5</sup> filter type
- 6-7: Reserved

Sinc<sup>x</sup> filter type transfer function: 
$$H(z) = \left( \frac{1 - z^{-FOSR}}{1 - z^{-1}} \right)^x$$

FastSinc filter type transfer function: 
$$H(z) = \left( \frac{1 - z^{-FOSR}}{1 - z^{-1}} \right)^2 \cdot (1 + z^{-(2 \cdot FOSR)})$$

This bit can only be modified when DFEN=0 (DFSDM\_FLTxCR1).

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

Bits 25:16 **FOSR[9:0]**: Sinc filter oversampling ratio (decimation rate)

0 - 1023: Defines the length of the Sinc type filter in the range 1 - 1024 (F<sub>OSR</sub> = FOSR[9:0] + 1). This number is also the decimation ratio of the output data rate from filter.

This bit can only be modified when DFEN=0 (DFSDM\_FLTxCR1)

*Note: If FOSR = 0, then the filter has no effect (filter bypass).*

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

Bits 7:0 **IOSR[7:0]**: Integrator oversampling ratio (averaging length)

0 - 255: The length of the Integrator in the range 1 - 256 (IOSR + 1). Defines how many samples from Sinc filter will be summed into one output data sample from the integrator. The output data rate from the integrator will be decreased by this number (additional data decimation ratio).

This bit can only be modified when DFEN=0 (DFSDM\_FLTxCR1)

*Note: If IOSR = 0, then the Integrator has no effect (Integrator bypass).*

### 27.8.7 DFSDM filter x data register for injected group (DFSDM\_FLTxJDATAR)

Address offset: 0x118 + 0x80 \* 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
JDATA[23:8]															
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
JDATA[7:0]								Res.	Res.	Res.	Res.	Res.	JDATA[2:0]		
r	r	r	r	r	r	r	r						r	r	r

Bits 31:8 **JDATA[23:0]**: Injected group conversion data

When each conversion of a channel in the injected group finishes, its resulting data is stored in this field. The data is valid when JEOCF=1. Reading this register clears the corresponding JEOCF.

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

Bits 2:0 **JDATA[2:0]**: Injected channel most recently converted

When each conversion of a channel in the injected group finishes, JDATA[2:0] is updated to indicate which channel was converted. Thus, JDATA[23:0] holds the data that corresponds to the channel indicated by JDATA[2:0].

*Note:* DMA may be used to read the data from this register. Half-word accesses may be used to read only the MSBs of conversion data.

Reading this register also clears JEOCF in DFSDM\_FLTxISR. Thus, the firmware must not read this register if DMA is activated to read data from this register.

### 27.8.8 DFSDM filter x data register for the regular channel (DFSDM\_FLTxRDATAR)

Address offset: 0x11C + 0x80 \* 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
RDATA[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA[7:0]								Res.	Res.	Res.	RPEND	Res.	RDATA[2:0]		
r	r	r	r	r	r	r	r				r		r	r	r

Bits 31:8 **RDATA[23:0]**: Regular channel conversion data

When each regular conversion finishes, its data is stored in this register. The data is valid when REOCF=1. Reading this register clears the corresponding REOCF.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **RPEND**: Regular channel pending data

Regular data in RDATA[23:0] was delayed due to an injected channel trigger during the conversion

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **RDATA[2:0]**: Regular channel most recently converted

When each regular conversion finishes, RDATA[2:0] is updated to indicate which channel was converted (because regular channel selection RCH[2:0] in DFSDM\_FLTxCR1 register can be updated during regular conversion). Thus RDATA[23:0] holds the data that corresponds to the channel indicated by RDATA[2:0].

*Note:* Half-word accesses may be used to read only the MSBs of conversion data.

Reading this register also clears REOCF in DFSDM\_FLTxISR.

### 27.8.9 DFSDM filter x analog watchdog high threshold register (DFSDM\_FLTxAWHTR)

Address offset: 0x120 + 0x80 \* 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
AWHT[23:8]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWHT[7:0]								Res.	Res.	Res.	Res.	BKAWH[3:0]			
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w					r/w	r/w	r/w	r/w

Bits 31:8 **AWHT[23:0]**: Analog watchdog high threshold

These bits are written by software to define the high threshold for the analog watchdog.

*Note: In case channel transceivers monitor (AWFSEL=1), the higher 16 bits (AWHT[23:8]) define the 16-bit threshold as compared with the analog watchdog filter output (because data coming from the analog watchdog filter are up to a 16-bit resolution). Bits AWHT[7:0] are not taken into comparison in this case.*

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

Bits 3:0 **BKAWH[3:0]**: Break signal assignment to analog watchdog high threshold event

BKAWH[i] = 0: Break i signal is not assigned to an analog watchdog high threshold event

BKAWH[i] = 1: Break i signal is assigned to an analog watchdog high threshold event

### 27.8.10 DFSDM filter x analog watchdog low threshold register (DFSDM\_FLTxAWLTR)

Address offset: 0x124 + 0x80 \* 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
AWLT[23:8]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWLT[7:0]								Res.	Res.	Res.	Res.	BKAWL[3:0]			
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w					r/w	r/w	r/w	r/w



Bits 31:8 **AWLT[23:0]**: Analog watchdog low threshold

These bits are written by software to define the low threshold for the analog watchdog.

*Note: In case channel transceivers monitor (AWFSEL=1), only the higher 16 bits (AWLT[23:8]) define the 16-bit threshold as compared with the analog watchdog filter output (because data coming from the analog watchdog filter are up to a 16-bit resolution). Bits AWLT[7:0] are not taken into comparison in this case.*

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

Bits 3:0 **BKAWL[3:0]**: Break signal assignment to analog watchdog low threshold event

BKAWL[i] = 0: Break i signal is not assigned to an analog watchdog low threshold event

BKAWL[i] = 1: Break i signal is assigned to an analog watchdog low threshold event

### 27.8.11 DFSDM filter x analog watchdog status register (DFSDM\_FLTxAWSR)

Address offset: 0x128 + 0x80 \* x, (x = 0 to 3)

Reset value: 0x0000 0000

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

AWHTF[y]=1 indicates a high threshold error on channel y. It is set by hardware. It can be cleared by software using the corresponding CLRAWHTF[y] bit in the DFSDM\_FLTxAWCFR register.

Bits 7:0 **AWLTF[7:0]**: Analog watchdog low threshold flag

AWLTF[y]=1 indicates a low threshold error on channel y. It is set by hardware. It can be cleared by software using the corresponding CLRAWLTF[y] bit in the DFSDM\_FLTxAWCFR register.

*Note: All the bits of DFSDM\_FLTxAWSR are automatically reset when DFEN=0.*

**27.8.12 DFSDM filter x analog watchdog clear flag register (DFSDM\_FLTxAWCFR)**

Address offset: 0x12C + 0x80 \* x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRAWHTF[7:0]								CLRAWLTF[7:0]							
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 31:16 Reserved, must be kept at reset value.

Bits 15:8 **CLRAWHTF[7:0]**: Clear the analog watchdog high threshold flag

CLRAWHTF[y]=0: Writing '0' has no effect

CLRAWHTF[y]=1: Writing '1' to position y clears the corresponding AWHTF[y] bit in the DFSDM\_FLTxAWSR register

Bits 7:0 **CLRAWLTF[7:0]**: Clear the analog watchdog low threshold flag

CLRAWLTF[y]=0: Writing '0' has no effect

CLRAWLTF[y]=1: Writing '1' to position y clears the corresponding AWLTF[y] bit in the DFSDM\_FLTxAWSR register

**27.8.13 DFSDM filter x extremes detector maximum register (DFSDM\_FLTxEXMAX)**

Address offset: 0x130 + 0x80 \* x, (x = 0 to 3)

Reset value: 0x8000 0000

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

Bits 31:8 **EXMAX[23:0]**: Extremes detector maximum value

These bits are set by hardware and indicate the highest value converted by DFSDM\_FLTx. EXMAX[23:0] bits are reset to value (0x800000) by reading of this register.

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

Bits 2:0 **EXMAXCH[2:0]**: Extremes detector maximum data channel.

These bits contains information about the channel on which the data is stored into EXMAX[23:0]. Bits are cleared by reading of this register.



**27.8.14 DFSDM filter x extremes detector minimum register (DFSDM\_FLTxEXMIN)**

Address offset: 0x134 + 0x80 \* x, (x = 0 to 3)

Reset value: 0x7FFF FF00

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

Bits 31:8 **EXMIN[23:0]**: Extremes detector minimum value

These bits are set by hardware and indicate the lowest value converted by DFSDM\_FLTx. EXMIN[23:0] bits are reset to value (0x7FFFFFFF) by reading of this register.

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

Bits 2:0 **EXMINCH[2:0]**: Extremes detector minimum data channel

These bits contain information about the channel on which the data is stored into EXMIN[23:0]. Bits are cleared by reading of this register.

**27.8.15 DFSDM filter x conversion timer register (DFSDM\_FLTxCNVTIMR)**

Address offset: 0x138 + 0x80 \* 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
CNVCNT[27:12]															
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
CNVCNT[11:0]												Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r				

Bits 31:4 **CNVCNT[27:0]**: 28-bit timer counting conversion time  $t = \text{CNVCNT}[27:0] / f_{\text{DFSDMCLK}}$

The timer has an input clock from DFSDM clock (system clock  $f_{\text{DFSDMCLK}}$ ). Conversion time measurement is started on each conversion start and stopped when conversion finishes (interval between first and last serial sample). Only in case of filter bypass ( $\text{FOSR}[9:0] = 0$ ) is the conversion time measurement stopped and  $\text{CNVCNT}[27:0] = 0$ . The counted time is:

if  $\text{FAST} = 0$  (or first conversion in continuous mode if  $\text{FAST} = 1$ ):

$$t = [\text{F}_{\text{OSR}} * (\text{I}_{\text{OSR}} - 1 + \text{F}_{\text{ORD}}) + \text{F}_{\text{ORD}}] / f_{\text{CKIN}} \dots \text{ for Sinc}^x \text{ filters}$$

$$t = [\text{F}_{\text{OSR}} * (\text{I}_{\text{OSR}} - 1 + 4) + 2] / f_{\text{CKIN}} \dots \text{ for FastSinc filter}$$

if  $\text{FAST} = 1$  in continuous mode (except first conversion):

$$t = [\text{F}_{\text{OSR}} * \text{I}_{\text{OSR}}] / f_{\text{CKIN}}$$

in case if  $\text{F}_{\text{OSR}} = \text{FOSR}[9:0] + 1 = 1$  (filter bypassed, active only integrator):

$$\text{CNVCNT} = 0 \text{ (counting is stopped, conversion time: } t = \text{I}_{\text{OSR}} / f_{\text{CKIN}})$$

where:  $f_{\text{CKIN}}$  is the channel input clock frequency (on given channel CKINy pin) or input data rate in case of parallel data input (from internal ADC or from CPU/DMA write)

*Note: When conversion is interrupted (e.g. by disable/enable selected channel) the timer counts also this interruption time.*

Bits 3:0 Reserved, must be kept at reset value.

### 27.8.16 DFSDM register map

The following table summarizes the DFSDM registers.

Table 169. DFSDM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	<b>DFSDM_CH0CFGR1</b>	DFSDMEN	CKOUTSRC	Res	Res	Res	Res	Res	Res	CKOUTDIV[7:0]							DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL [1:0]		SITP[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
0x04	<b>DFSDM_CH0CFGR2</b>	OFFSET[23:0]																							DTRBS[4:0]				Res	Res	Res			
	reset value	0																							0									
0x08	<b>DFSDM_CH0AWSCDR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0
0x0C	<b>DFSDM_CH0WDATR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	<b>DFSDM_CH0DAT1NR</b>	INDAT1[15:0]															INDAT0[15:0]																	
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x14	<b>DFSDM_CH0DLYR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	reset value																											0	0	0	0	0	0	0
0x18 - 0x1C	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.	

Table 169. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	DFSDM_CH1CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0	0	0	0	0	0
0x24	DFSDM_CH1CFGR2	OFFSET[23:0]																							DTRBS[4:0]				Res.	Res.	Res.		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x28	DFSDM_CH1AWSCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	reset value																																
0x2C	DFSDM_CH1WDATR	WDATA[15:0]																							Res.	Res.	Res.						
	reset value																																
0x30	DFSDM_CH1DATINR	INDAT1[15:0]											INDAT0[15:0]																				
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x34	DFSDM_CH1DLR	Res.	Res.	Res.	Res.	Res.	Res.	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 - 0x3C	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.		
0x40	DFSDM_CH2CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]
	reset value																									0	0	0	0	0	0	0	0
0x44	DFSDM_CH2CFGR2	OFFSET[23:0]																							DTRBS[4:0]				Res.	Res.	Res.		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x48	DFSDM_CH2AWSCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	reset value																																
0x4C	DFSDM_CH2WDATR	WDATA[15:0]																							Res.	Res.	Res.						
	reset value																																
0x50	DFSDM_CH2DATINR	INDAT1[15:0]											INDAT0[15:0]																				
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x54	DFSDM_CH2DLR	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0x58 - 0x5C	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.		



Table 169. DFSDM 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
0x60	DFSDM_CH3CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]	Res.	DATMPX[1:0]	Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]	Res.	SITP[1:0]		
	reset value																	0	0	0	0				0	0	0	0	0	0	0	0	0
0x64	DFSDM_CH3CFGR2	OFFSET[23:0]																							DTRBS[4:0]				Res.	Res.	Res.		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x68	DFSDM_CH3AWSCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0x6C	DFSDM_CH3WDATR	WDATA[15:0]																															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x70	DFSDM_CH3DATINR	INDAT1[15:0]															INDAT0[15:0]																
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x74	DFSDM_CH3DLYR	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0x78 - 0x7C	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.		
0x80	DFSDM_CH4CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]	Res.	DATMPX[1:0]	Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]	Res.	SITP[1:0]	
	reset value																																
0x84	DFSDM_CH4CFGR2	OFFSET[23:0]																							DTRBS[4:0]				Res.	Res.	Res.		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x88	DFSDM_CH4AWSCDR	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0x8C	DFSDM_CH4WDATR	WDATA[15:0]																															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x90	DFSDM_CH4DATINR	INDAT1[15:0]															INDAT0[15:0]																
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x94	DFSDM_CH4DLYR	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0x98 - 0x9C	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.		



Table 169. DFSDM 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	
0xA0	DFSDM_CH5CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]	DATMPX[1:0]	Res.	Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]	Res.	Res.	SITP[1:0]	Res.	
	reset value																	0	0	0	0					0	0	0	0	0	0	0	0	0
0xA4	DFSDM_CH5CFGR2	OFFSET[23:0]																								DTRBS[4:0]				Res.	Res.	Res.		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0xA8	DFSDM_CH5AWSCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																	
0xAC	DFSDM_CH5WDATR	WDATA[15:0]																																
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0xB0	DFSDM_CH5DATINR	INDAT1[15:0]															INDAT0[15:0]																	
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0xB4	DFSDM_CH5DLYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																	
0xB8 - 0xBC	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.		
0xC0	DFSDM_CH6CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]	DATMPX[1:0]	Res.	Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]	Res.	Res.	SITP[1:0]	Res.
	reset value																																	
0xC4	DFSDM_CH6CFGR2	OFFSET[23:0]																								DTRBS[4:0]				Res.	Res.	Res.		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0xC8	DFSDM_CH6AWSCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	reset value																																	
0xCC	DFSDM_CH6WDATR	WDATA[15:0]																																
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0xD0	DFSDM_CH6DATINR	INDAT1[15:0]															INDAT0[15:0]																	
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0xD4	DFSDM_CH6DLYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	reset value																																	
0xD8 - 0xDC	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.	

Table 169. DFSDM 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				
0xE0	DFSDM_CH7CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]	DATMPX[1:0]	Res.	Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPIKSEL[1:0]	Res.	Res.	SITP[1:0]					
	reset value																	0	0	0	0					0	0	0	0	0	0	0	0	0			
0xE4	DFSDM_CH7CFGR2	OFFSET[23:0]																								DTRBS[4:0]											
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0xE8	DFSDM_CH7AWSCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
	reset value																																				
0xEC	DFSDM_CH7WDATR	WDATA[15:0]																																			
	reset value																																				
0xF0	DFSDM_CH7DATINR	INDAT1[15:0]															INDAT0[15:0]																				
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0xF4	DFSDM_CH7DLYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	reset value																																				
0xF8 - 0xFC	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.			
0x100	DFSDM_FLT0CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																				
0x104	DFSDM_FLT0CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																				
0x108	DFSDM_FLT0ISR	SCDF[7:0]							CKABF[7:0]							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	
0x10C	DFSDM_FLT0ICR	CLRSCDF[7:0]							CLRCKABF[7:0]							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
0x110	DFSDM_FLT0JCHGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																				
0x114	DFSDM_FLT0FCR	FORD[2:0]		Res.	Res.	Res.	FOSR[9:0]									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																																	





Table 169. DFSDM 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																								
0x118	DFSDM_FLT0JDATAR	JDATA[23:0]																								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	
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																						
0x11C	DFSDM_FLT0RDATAR	RDATA[23:0]																								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
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																						
0x120	DFSDM_FLT0AWHTR	AWHT[23:0]																								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
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																						
0x124	DFSDM_FLT0AWLTR	AWLT[23:0]																								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
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																						
0x128	DFSDM_FLT0AWSR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	AWHTF[7:0]							AWLTF[7:0]																																
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																							
0x12C	DFSDM_FLT0AWCFR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CLRAWHTF[7:0]							CLRAWLTF[7:0]																																
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																							
0x130	DFSDM_FLT0EXMAX	EXMAX[23:0]																								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
	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																							
0x134	DFSDM_FLT0EXMIN	EXMIN[23:0]																								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
	reset value	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	1	1																							
0x138	DFSDM_FLT0CNVTIMR	CNVCNT[27:0]																								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
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																							
0x13C-0x17C	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																							
0x180	DFSDM_FLT1CR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	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																							
0x184	DFSDM_FLT1CR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	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																							
0x188	DFSDM_FLT1ISR	Res	Res	Res	Res	Res	Res	Res	Res	Res	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																							



Table 169. DFSDM 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																					
0x18C	DFSDM_FLT1ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																						
0x190	DFSDM_FLT1JCHGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																				
0x194	DFSDM_FLT1FCR	FORD[2:0]		Res.	Res.	Res.	FOSR[9:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[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																					
0x198	DFSDM_FLT1JDATAR	JDATA[23:0]																							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.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																					
0x19C	DFSDM_FLT1RDATAR	RDATA[23:0]																							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.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																					
0x1A0	DFSDM_FLT1AWHTR	AWHT[23:0]																							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.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																					
0x1A4	DFSDM_FLT1AWLTR	AWLT[23:0]																							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.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																					
0x1A8	DFSDM_FLT1AWSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWHTF[7:0]					AWLTF[7:0]																														
	reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0																						
0x1AC	DFSDM_FLT1AWCFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRAWHTF[7:0]					CLRAWLTF[7:0]																														
	reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0																						
0x1B0	DFSDM_FLT1EXMAX	EXMAX[23:0]																							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.
	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																						
0x1B4	DFSDM_FLT1EXMIN	EXMIN[23:0]																							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.
	reset value	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	0																						
0x1B8	DFSDM_FLT1CNVTIMR	CNVCNT[27:0]																																																				
	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																							
0x1BC - 0x1FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																						



Table 169. DFSDM 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																									
0x200	DFSDM_FLT2CR1	Res.	AWFSEL	FAST	Res.	Res.	RCH[2:0]			Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSW START	Res.	Res.	JEXTEN[1:0]			JEXTSEL[4:0]				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																								
0x204	DFSDM_FLT2CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[7:0]							EXCH[7:0]							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																								
0x208	DFSDM_FLT2ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCIP	JCIP	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																								
0x20C	DFSDM_FLT2ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																								
0x210	DFSDM_FLT2JCHGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[7:0]				Res.	Res.	Res.	Res.																								
	reset value																										0	0	0	0	0	0	0	0	1																							
0x214	DFSDM_FLT2FCR	FORD[2:0]		Res.	Res.	Res.	FOSR[9:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[7:0]				Res.	Res.	Res.	Res.																									
	reset value	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0																							
0x218	DFSDM_FLT2JDATAR	JDATA[23:0]																							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.	JDATA[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																						
0x21C	DFSDM_FLT2RDATAR	RDATA[23:0]																							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.	RDATA[2:0]	
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																					
0x220	DFSDM_FLT2AWHTR	AWHT[23:0]																							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.	BKAWH[3:0]	
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																					
0x224	DFSDM_FLT2AWLTR	AWLT[23:0]																							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.	Res.	BKAWL[3:0]
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																					
0x228	DFSDM_FLT2AWSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWHTF[7:0]				AWLTF[7:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																				
	reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																						
0x22C	DFSDM_FLT2AWCFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRAWHTF[7:0]				CLRAWLTF[7:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																				
	reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																						

Table 169. DFSDM 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																									
0x230	DFSDM_FLT2EXMAX	EXMAX[23:0]																								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	
	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																					
0x234	DFSDM_FLT2EXMIN	EXMIN[23:0]																								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	Res
	reset value	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	1	1	1	1	1	1																				
0x238	DFSDM_FLT2CNVTMR	CNVCNT[27:0]																								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	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																				
0x23C - 0x27C	Reserved	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res																					
0x280	DFSDM_FLT3CR1	Res	Res	AWFSEL	FAST	Res	Res	Res	RCH[2:0]	Res	Res	RDMAEN	Res	RSYNC	RCONT	Res	Res	Res	JEXTEN[1:0]	Res	JEXTSEL[4:0]	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																					
0x284	DFSDM_FLT3CR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	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																					
0x288	DFSDM_FLT3ISR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RCIP	JCIP	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																					
0x28C	DFSDM_FLT3ICR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	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																					
0x290	DFSDM_FLT3JCHGR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	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	1																					
0x294	DFSDM_FLT3FCR	FORD[2:0]	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	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																					
0x298	DFSDM_FLT3JDATAR	JDATA[23:0]																								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	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																				



Table 169. DFSDM 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
0x29C	DFSDM_FLT3RDATAR	RDATA[23:0]																								Res	Res	Res	Res	RDATA CH[2:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2A0	DFSDM_FLT3AWHTR	AWHT[23:0]																								Res	Res	Res	Res	BKAWH[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2A4	DFSDM_FLT3AWLTR	AWLT[23:0]																								Res	Res	Res	Res	BKAWL[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2A8	DFSDM_FLT3AWSR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	AWHTF[7:0]				AWLTF[7:0]											
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2AC	DFSDM_FLT3AWCFR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CLRAWHTF[7:0]				CLRAWLTF[7:0]											
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2B0	DFSDM_FLT3EXMAX	EXMAX[23:0]																								Res	Res	Res	Res	EXMAXCH[2:0]			
	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
0x2B4	DFSDM_FLT3EXMIN	EXMIN[23:0]																								Res	Res	Res	Res	EXMINCH[2:0]			
	reset value	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	0
0x2B8	DFSDM_FLT3CNVTIMR	CNVCNT[27:0]																								Res	Res	Res	Res				
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2BC - 0x3FC	Reserved	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.

## 28 LCD-TFT display controller (LTDC)

### 28.1 Introduction

The LCD-TFT (liquid crystal display - thin film transistor) display controller provides a parallel digital RGB (red, green, blue) and signals for horizontal, vertical synchronization, pixel clock and data enable as output to interface directly to a variety of LCD and TFT panels.

### 28.2 LTDC main features

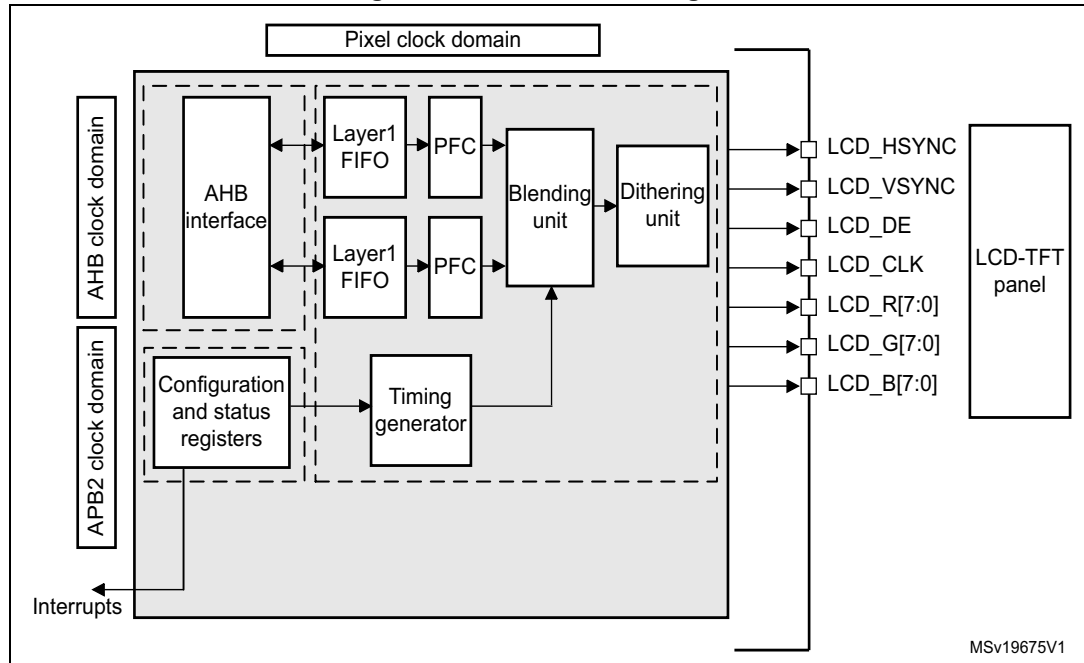
- 24-bit RGB parallel pixel output; 8 bits-per-pixel (RGB888)
- 2 display layers with dedicated FIFO (64x32-bit)
- Color look-up table (CLUT) up to 256 color (256x24-bit) per layer
- Programmable timings for different display panels
- Programmable background color
- Programmable polarity for HSYNC, VSYNC and data enable
- Up to 8 input color formats selectable per layer
  - ARGB8888
  - RGB888
  - RGB565
  - ARGB1555
  - ARGB4444
  - L8 (8-bit luminance or CLUT)
  - AL44 (4-bit alpha + 4-bit luminance)
  - AL88 (8-bit alpha + 8-bit luminance)
- Pseudo-random dithering output for low bits per channel
  - Dither width 2-bits for red, green, blue
- Flexible blending between two layers using alpha value (per pixel or constant)
- Color keying (transparency color)
- Programmable window position and size
- Supports thin film transistor (TFT) color displays
- AHB master interface with burst of 16 words
- Up to 4 programmable interrupt events

## 28.3 LTDC functional description

### 28.3.1 LTDC block diagram

The block diagram of the LTDC is shown in [Figure 171: LTDC block diagram](#).

Figure 171. LTDC block diagram



Layer FIFO: One FIFO 64x32-bit per layer.

PFC: pixel format converter, performing the pixel format conversion from the selected input pixel format of a layer to words.

AHB interface: for data transfer from memories to the FIFO.

Blending, dithering unit and timings generator: Refer to [Section 28.4.1](#) and [Section 28.4.2](#).

### 28.3.2 LTDC pins and external signal interface

[Table 170](#) summarizes the LTDC signal interface.

Table 170. LTDC pins and signal interface

LCD-TFT signals	I/O	Description
LCD_CLK	O	Clock output
LCD_HSYNC	O	Horizontal synchronization
LCD_VSYNC	O	Vertical synchronization
LCD_DE	O	Not data enable
LCD_R[7:0]	O	Data: 8-bit red data

**Table 170. LTDC pins and signal interface (continued)**

LCD-TFT signals	I/O	Description
LCD_G[7:0]	O	Data: 8-bit green data
LCD_B[7:0]	O	Data: 8-bit blue data

The LTDC-TFT controller pins must be configured by the user application. The unused pins can be used for other purposes.

For LTDC outputs up to 24-bit (RGB888), if less than 8 bpp are used to output for example RGB565 or RGB666 to interface on 16- or 18-bit displays, the RGB display data lines must be connected to the MSB of the LCD-TFT controller RGB data lines. As an example, in the case of an LCD-TFT controller interfacing with a RGB565 16-bit display, the LCD display R[4:0], G[5:0] and B[4:0] data lines pins must be connected to LCD-TFT controller LCD\_R[7:3], LCD\_G[7:2] and LCD\_B[7:3].

### 28.3.3 LTDC reset and clocks

The LCD-TFT controller peripheral uses 3 clock domains:

- AHB clock domain (HCLK)
  - This domain contains the LCD-TFT AHB master interface for data transfer from the memories to the Layer FIFO and the frame buffer configuration register
- APB2 clock domain (PCLK2):
  - This domain contains the global configuration registers and the interrupt register.
- Pixel clock domain (LCD\_CLK)
  - This domain contains the pixel data generation, the layer configuration register as well as the LCD-TFT interface signal generator. The LCD\_CLK output should be configured following the panel requirements. The LCD\_CLK is generated from a specific PLL output (refer to the reset and clock control section).

*Table 171* summarizes the clock domain for each register.

**Table 171. Clock domain for each register**

LTDC register	Clock domain
LTDC_LxCR	HCLK
LTDC_LxCFBAR	
LTDC_LxCFBLR	
LTDC_LxCFBLNR	
LTDC_SRCR	PCLK2
LTDC_IER	
LTDC_ISR	
LTDC_ICR	



**Table 171. Clock domain for each register (continued)**

LTDC register	Clock domain
LTDC_SSCR	Pixel clock (LCD_CLK)
LTDC_BPCR	
LTDC_AWCR	
LTDC_TWCR	
LTDC_GCR	
LTDC_BCCR	
LTDC_LIPCR	
LTDC_CPSR	
LTDC_CDSR	
LTDC_LxWHPCR	
LTDC_LxWVPCR	
LTDC_LxCKCR	
LTDC_LxPFCR	
LTDC_LxCACR	
LTDC_LxDCCR	
LTDC_LxBFCR	
LTDC_LxCLUTWR	

Care must be taken while accessing the LTDC registers, the APB2 bus is stalled during:

- 6 PCKL2 periods + 5 LCD\_CLK periods (5 HCLK periods for register on AHB clock domain) for register write access and update;
- 7 PCKL2 periods + 5 LCD\_CLK periods (5 HCLK periods for register on AHB clock domain) for register read access.

For registers on PCKL2 clock domain, APB2 bus is stalled for 6 PCKL2 periods during the register write accesses, and for 7 PCKL2 periods during Read accesses.

The LCD controller can be reset by setting the corresponding bit in the RCC\_APB2RSTR register. It resets the three clock domains.

## 28.4 LTDC programmable parameters

The LCD-TFT controller provides flexible configurable parameters. It can be enabled or disabled through the LTDC\_GCR register.

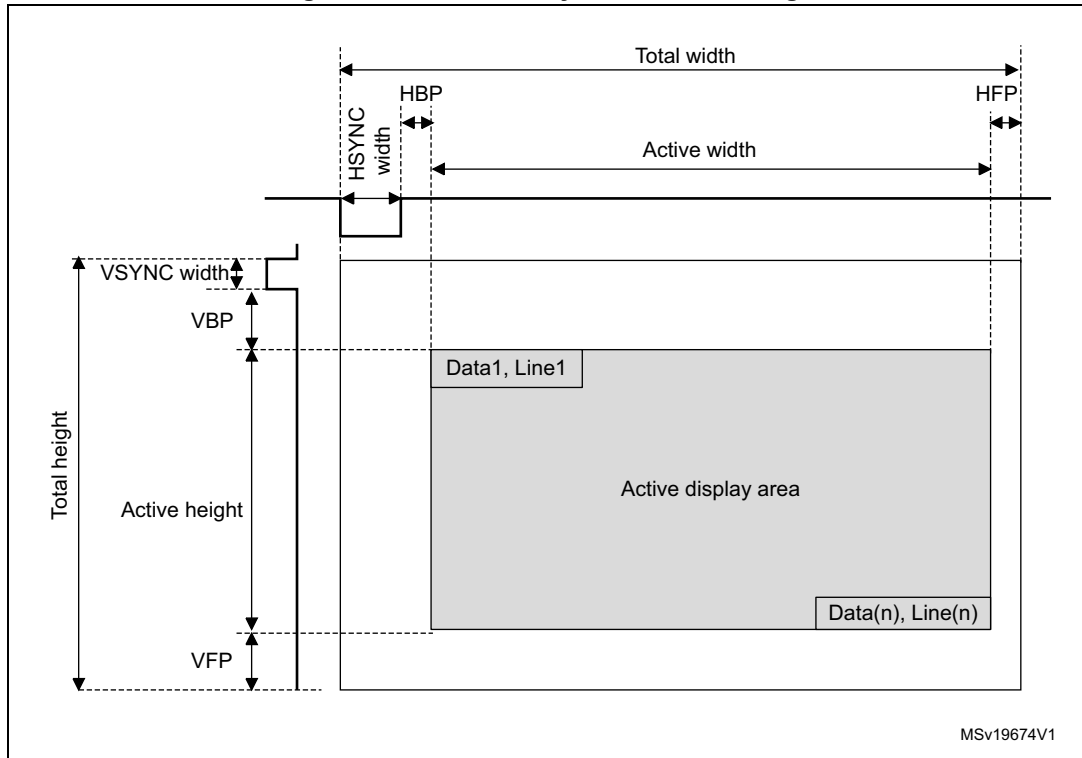
### 28.4.1 LTDC global configuration parameters

#### Synchronous timings

*Figure 172* presents the configurable timing parameters generated by the synchronous timings generator block presented in the block diagram *Figure 171*. It generates the

horizontal and vertical synchronization timings panel signals, the pixel clock and the data enable signals.

Figure 172. LCD-TFT synchronous timings



Note: The HBP and HFP are respectively the horizontal back porch and front porch period. The VBP and the VFP are respectively the vertical back porch and front porch period.

The LCD-TFT programmable synchronous timings are:

- HSYNC and VSYNC width: horizontal and vertical synchronization width, configured by programming a value of **HSYNC width - 1** and **VSYNC width - 1** in the LTDC\_SSCR register
- HBP and VBP: horizontal and vertical synchronization back porch width, configured by programming the accumulated value **HSYNC width + HBP - 1** and the accumulated value **VSYNC width + VBP - 1** in the LTDC\_BPCR register.
- Active width and active height: the active width and active height are configured by programming the accumulated value **HSYNC width + HBP + active width - 1** and the accumulated value **VSYNC width + VBP + active height - 1** in the LTDC\_AWCR register (only up to 1024x768 is supported).
- Total width: the total width is configured by programming the accumulated value **HSYNC width + HBP + active width + HFP - 1** in the LTDC\_TWCR register. The HFP is the horizontal front porch period.
- Total height: the total height is configured by programming the accumulated value **VSYNC height + VBP + active height + VFP - 1** in the LTDC\_TWCR register. The VFP is the vertical front porch period.

*Note:* When the LTDC is enabled, the timings generated start with X/Y=0/0 position as the first horizontal synchronization pixel in the vertical synchronization area and following the back porch, active data display area and the front porch.

When the LTDC is disabled, the timing generator block is reset to  $X = total\ width - 1$ ,  $Y = total\ height - 1$  and held the last pixel before the vertical synchronization phase and the FIFO are flushed. Therefore only blanking data is output continuously.

### Example of synchronous timings configuration

TFT-LCD timings (must be extracted from panel datasheet):

- horizontal and vertical synchronization width: 0xA pixels and 0x2 lines
- horizontal and vertical back porch: 0x14 pixels and 0x2 lines
- active width and active height: 0x140 pixels, 0xF0 lines (320x240)
- horizontal front porch: 0xA pixels
- vertical front porch: 0x4 lines

The programmed values in the LTDC timings registers are:

- LTDC\_BPCR register: to be programmed to 0x001D0003 (AHBP[11:0] is 0x1D(0xA+0x13) and AVBP[10:0] is 0x3(0x2 + 0x1)).
- LTDC\_AWCR register: to be programmed to 0x015D00F3 (AAW[11:0] is 0x15D(0xA+0x14 +0x13F) and AAH[10:0] is 0xF3(0x2 +0x2 + 0xEF).
- LTDC\_TWCR register: to be programmed to 0x00000167 (TOTALW[11:0] is 0x167(0xA+0x14 +0x140 + 0x9).
- LTDC\_THCR register: to be programmed to 0x000000F7 (TOTALH[10:0] is 0xF7(0x2 +0x2 + 0xF0 + 3)

### Programmable polarity

The horizontal and vertical synchronization, data enable and pixel clock output signals polarity can be programmed to active high or active low through the LTDC\_GCR register.

### Background color

A constant background color (RGB888) can be programmed through the LTDC\_BCCR register. It is used for blending with the bottom layer.

### Dithering

The dithering pseudo-random technique using an LFSR is used to add a small random value (threshold) to each pixel color channel (R, G or B) value, thus rounding up the MSB in some cases when displaying a 24-bit data on 18-bit display. Thus the Dithering technique is used to round data which is different from one frame to the other.

The dithering pseudo-random technique is the same as comparing LSBs against a threshold value and adding a 1 to the MSB part only, if the LSB part is  $\geq$  the threshold. The LSBs are typically dropped once dithering was applied.

The width of the added pseudo-random value is 2 bits for each color channel: 2 bits for red, 2 bits for green and 2 bits for blue.

Once the LCD-TFT controller is enabled, the LFSR starts running with the first active pixel and it is kept running even during blanking periods and when dithering is switched off. If the LTDC is disabled, the LFSR is reset.

The dithering can be switched On and Off on the fly through the LTDC\_GCR register.

### Reload shadow registers

Some configuration registers are shadowed. The shadow registers values can be reloaded immediately to the active registers when writing to these registers or at the beginning of the vertical blanking period following the configuration in the LTDC\_SRCR register. If the immediate reload configuration is selected, the reload should be only activated when all new registers have been written.

The shadow registers should not be modified again before the reload has been done. Reading from the shadow registers returns the actual active value. The new written value can only be read after the reload has taken place.

A register reload interrupt can be generated if enabled in the LTDC\_IER register.

The shadowed registers are all the layer1 and layer2 registers except the LTDC\_LxCLUTWR register.

### Interrupt generation event

Refer to [Section 28.5: LTDC interrupts](#) for interrupt configuration.

## 28.4.2 Layer programmable parameters

Up to two layers can be enabled, disabled and configured separately. The layer display order is fixed and it is bottom up. If two layers are enabled, the layer2 is the top displayed window.

### Windowing

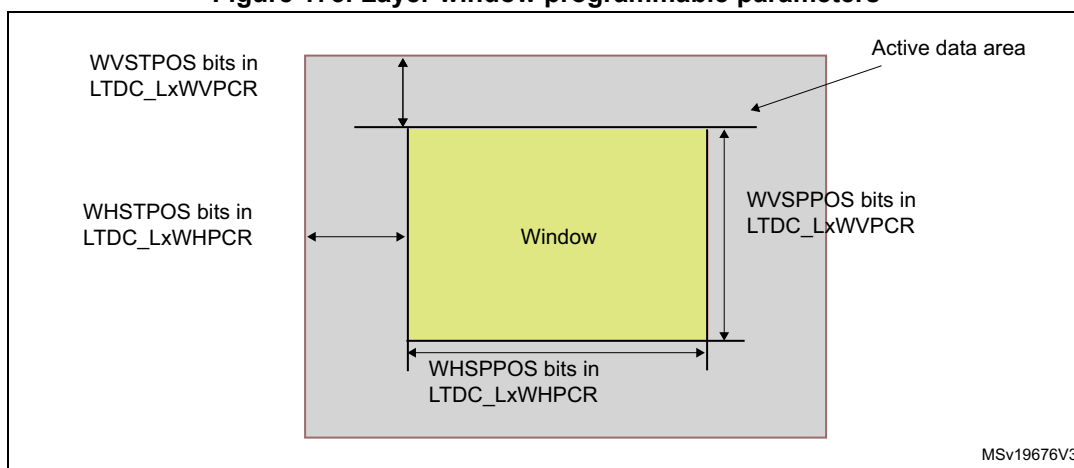
Every layer can be positioned and resized and it must be inside the active display area.

The window position and size are configured through the top-left and bottom-right X/Y positions and the internal timing generator that includes the synchronous, back porch size and the active data area. Refer to LTDC\_LxWHPCR and LTDC\_WVPCR registers.

The programmable layer position and size defines the first/last visible pixel of a line and the first/last visible line in the window. It allows to display either the full image frame or only a part of the image frame. Refer to [Figure 173](#).

- The first and the last visible pixel in the layer are set by configuring the WHSTPOS[11:0] and WHSPPOS[11:0] in the LTDC\_LxWHPCR register.
- The first and the last visible lines in the layer are set by configuring the WVSTPOS[10:0] and WVSPPOS[10:0] in the LTDC\_LxWVPCR register.

Figure 173. Layer window programmable parameters



**Pixel input format**

The programmable pixel format is used for the data stored in the frame buffer of a layer.

Up to 8 input pixel formats can be configured for every layer through the LTDC\_LxPFCR register

The pixel data is read from the frame buffer and then transformed to the internal 8888 (ARGB) format as follows: components having a width of less than 8 bits get expanded to 8 bits by bit replication. The selected bit range is concatenated multiple times until it is longer than 8 bits. Of the resulting vector, the 8 MSB bits are chosen. Example: 5 bits of an RGB565 red channel become (bit positions): 43210432 (the 3 LSBs are filled with the 3 MSBs of the 5 bits)

Table 172 describes the pixel data mapping depending on the selected format.

Table 172. Pixel data mapping versus color format

ARGB8888			
@+3 A <sub>x</sub> [7:0]	@+2 R <sub>x</sub> [7:0]	@+1 G <sub>x</sub> [7:0]	@ B <sub>x</sub> [7:0]
@+7 A <sub>x+1</sub> [7:0]	@+6 R <sub>x+1</sub> [7:0]	@+5 G <sub>x+1</sub> [7:0]	@+4 B <sub>x+1</sub> [7:0]
RGB888			
@+3 B <sub>x+1</sub> [7:0]	@+2 R <sub>x</sub> [7:0]	@+1 G <sub>x</sub> [7:0]	@ B <sub>x</sub> [7:0]
@+7 G <sub>x+2</sub> [7:0]	@+6 B <sub>x+2</sub> [7:0]	@+5 R <sub>x+1</sub> [7:0]	@+4 G <sub>x+1</sub> [7:0]
RGB565			
@+3 R <sub>x+1</sub> [4:0] G <sub>x+1</sub> [5:3]	@+2 G <sub>x+1</sub> [2:0] B <sub>x+1</sub> [4:0]	@+1 R <sub>x</sub> [4:0] G <sub>x</sub> [5:3]	@ G <sub>x</sub> [2:0] B <sub>x</sub> [4:0]
@+7 R <sub>x+3</sub> [4:0] G <sub>x+3</sub> [5:3]	@+6 G <sub>x+3</sub> [2:0] B <sub>x+3</sub> [4:0]	@+5 R <sub>x+2</sub> [4:0] G <sub>x+2</sub> [5:3]	@+4 G <sub>x+2</sub> [2:0] B <sub>x+2</sub> [4:0]
ARGB1555			

**Table 172. Pixel data mapping versus color format (continued)**

@+3 A <sub>x+1</sub> [0]R <sub>x+1</sub> [4:0] G <sub>x+1</sub> [4:3]	@+2 G <sub>x+1</sub> [2:0] B <sub>x+1</sub> [4:0]	@+1 A <sub>x</sub> [0] R <sub>x</sub> [4:0] G <sub>x</sub> [4:3]	@ G <sub>x</sub> [2:0] B <sub>x</sub> [4:0]
@+7 A <sub>x+3</sub> [0]R <sub>x+3</sub> [4:0] G <sub>x+3</sub> [4:3]	@+6 G <sub>x+3</sub> [2:0] B <sub>x+3</sub> [4:0]	@+5 A <sub>x+2</sub> [0]R <sub>x+2</sub> [4:0]G <sub>x+2</sub> [4:3]	@+4 G <sub>x+2</sub> [2:0] B <sub>x+2</sub> [4:0]
<b>ARGB4444</b>			
@+3 A <sub>x+1</sub> [3:0]R <sub>x+1</sub> [3:0]	@+2 G <sub>x+1</sub> [3:0] B <sub>x+1</sub> [3:0]	@+1 A <sub>x</sub> [3:0] R <sub>x</sub> [3:0]	@ G <sub>x</sub> [3:0] B <sub>x</sub> [3:0]
@+7 A <sub>x+3</sub> [3:0]R <sub>x+3</sub> [3:0]	@+6 G <sub>x+3</sub> [3:0] B <sub>x+3</sub> [3:0]	@+5 A <sub>x+2</sub> [3:0]R <sub>x+2</sub> [3:0]	@+4 G <sub>x+2</sub> [3:0] B <sub>x+2</sub> [3:0]
<b>L8</b>			
@+3 L <sub>x+3</sub> [7:0]	@+2 L <sub>x+2</sub> [7:0]	@+1 L <sub>x+1</sub> [7:0]	@ L <sub>x</sub> [7:0]
@+7 L <sub>x+7</sub> [7:0]	@+6 L <sub>x+6</sub> [7:0]	@+5 L <sub>x+5</sub> [7:0]	@+4 L <sub>x+4</sub> [7:0]
<b>AL44</b>			
@+3 A <sub>x+3</sub> [3:0] L <sub>x+3</sub> [3:0]	@+2 A <sub>x+2</sub> [3:0] L <sub>x+2</sub> [3:0]	@+1 A <sub>x+1</sub> [3:0] L <sub>x+1</sub> [3:0]	@ A <sub>x</sub> [3:0] L <sub>x</sub> [3:0]
@+7 A <sub>x+7</sub> [3:0] L <sub>x+7</sub> [3:0]	@+6 A <sub>x+6</sub> [3:0] L <sub>x+6</sub> [3:0]	@+5 A <sub>x+5</sub> [3:0] L <sub>x+5</sub> [3:0]	@+4 A <sub>x+4</sub> [3:0] L <sub>x+4</sub> [3:0]
<b>AL88</b>			
@+3 A <sub>x+1</sub> [7:0]	@+2 L <sub>x+1</sub> [7:0]	@+1 A <sub>x</sub> [7:0]	@ L <sub>x</sub> [7:0]
@+7 A <sub>x+3</sub> [7:0]	@+6 L <sub>x+3</sub> [7:0]	@+5 A <sub>x+2</sub> [7:0]	@+4 L <sub>x+2</sub> [7:0]

**Color look-up table (CLUT)**

The CLUT can be enabled at run-time for every layer through the LTDC\_LxCR register and it is only useful in case of indexed color when using the L8, AL44 and AL88 input pixel format.

First, the CLUT has to be loaded with the R, G and B values that replace the original R, G, B values of that pixel (indexed color). Each color (RGB value) has its own address which is the position within the CLUT.

The R, G and B values and their own respective address are programmed through the LTDC\_LxCLUTWR register.

- In case of L8 and AL88 input pixel format, the CLUT has to be loaded by 256 colors. The address of each color is configured in the CLUTADD bits in the LTDC\_LxCLUTWR register.
- In case of AL44 input pixel format, the CLUT has to be only loaded by 16 colors. The address of each color must be filled by replicating the 4-bit L channel to 8-bit as follows:

- L0 (indexed color 0), at address 0x00
- L1, at address 0x11
- L2, at address 0x22
- .....
- L15, at address 0xFF

**Color frame buffer address**

Every layer has a start address for the color frame buffer configured through the LTDC\_LxCFBAR register.

When a layer is enabled, the data is fetched from the color frame buffer.

**Color frame buffer length**

Every layer has a total line length setting for the color frame buffer in bytes and a number of lines in the frame buffer configurable in the LTDC\_LxCFBLR and LTDC\_LxCFBLNR register respectively.

The line length and the number of lines settings are used to stop the prefetching of data to the layer FIFO at the end of the frame buffer.

- If it is set to less bytes than required, a FIFO underrun interrupt is generated if it has been previously enabled.
- If it is set to more bytes than actually required, the useless data read from the FIFO is discarded. The useless data is not displayed.

**Color frame buffer pitch**

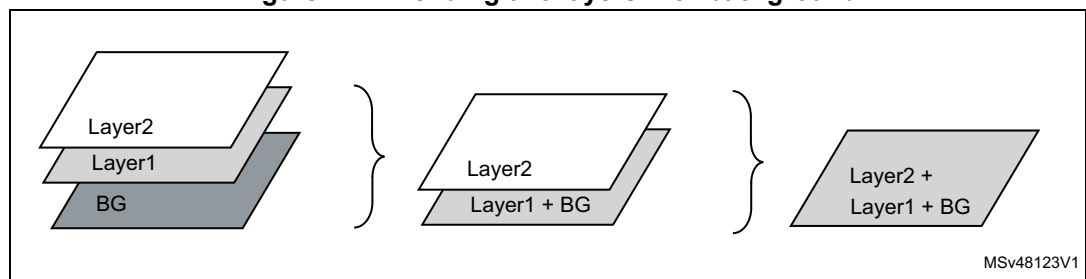
Every layer has a configurable pitch for the color frame buffer, which is the distance between the start of one line and the beginning of the next line in bytes. It is configured through the LTDC\_LxCFBLR register.

**Layer blending**

The blending is always active and the two layers can be blended following the blending factors configured through the LTDC\_LxBFCR register.

The blending order is fixed and it is bottom up. If two layers are enabled, first the Layer1 is blended with the Background color, then the layer2 is blended with the result of blended color of layer1 and the background. Refer to [Figure 174](#).

**Figure 174. Blending two layers with background**



MSv48123V1

### Default color

Every layer can have a default color in the format ARGB which is used outside the defined layer window or when a layer is disabled.

The default color is configured through the LTDC\_LxDCCR register.

The blending is always performed between the two layers even when a layer is disabled. To avoid displaying the default color when a layer is disabled, keep the blending factors of this layer in the LTDC\_LxBFCR register to their reset value.

### Color keying

A color key (RGB) can be configured to be representative for a transparent pixel.

If the color keying is enabled, the current pixels (after format conversion and before CLUT respectively blending) are compared to the color key. If they match for the programmed RGB value, all channels (ARGB) of that pixel are set to 0.

The color key value can be configured and used at run-time to replace the pixel RGB value.

The color keying is enabled through the LTDC\_LxCKCR register.

The color keying is configured through the LTDC\_LxCKCR register. The programmed value depends on the pixel format as it is compared to current pixel after pixel format conversion to ARGB888.

Example: if the a mid-yellow color (50% red + 50% green) is used as the transparent color key:

- In RGB565, the mid-yellow color is 0x8400. Set the LTDC\_LxCKCR to 0x848200.
- In ARGB8888, the mid-yellow color is 0x808000, set LTDC\_LxCKCR to 0x808000.
- In all CLUT-based color modes (L8, AL88, AL44), set one of the palette entry to the mid-yellow color 0x808000 and set the LTDC\_LxCKCR to 0x808000.

## 28.5 LTDC interrupts

The LTDC provides four maskable interrupts logically ORed to two interrupt vectors.

The interrupt sources can be enabled or disabled separately through the LTDC\_IER register. Setting the appropriate mask bit to 1 enables the corresponding interrupt.

The two interrupts are generated on the following events:

- Line interrupt: generated when a programmed line is reached. The line interrupt position is programmed in the LTDC\_LIPCR register
- Register reload interrupt: generated when the shadow registers reload was performed during the vertical blanking period
- FIFO underrun interrupt: generated when a pixel is requested from an empty layer FIFO
- Transfer error interrupt: generated when an AHB bus error occurs during data transfer

Those interrupts events are connected to the NVIC controller as described in [Figure 175](#).



Figure 175. Interrupt events

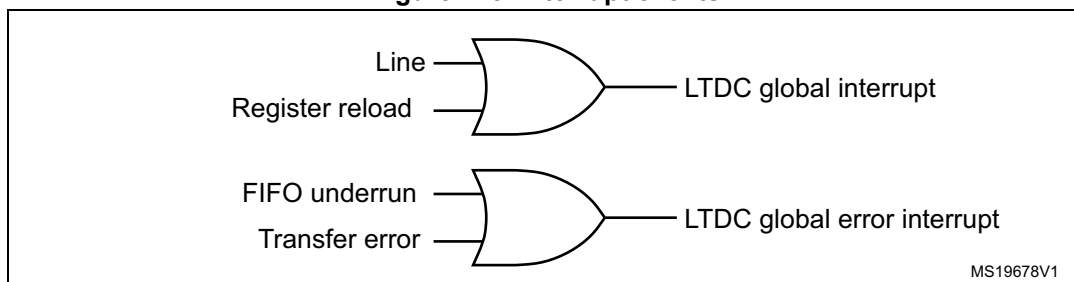


Table 173. LTDC interrupt requests

Interrupt event	Event flag	Enable control bit
Line	LIF	LIE
Register reload	RRIF	RRIEN
FIFO underrun	FUDERRIF	FUDERRIE
Transfer error	TERRIF	TERRIE

## 28.6 LTDC programming procedure

- Enable the LTDC clock in the RCC register.
- Configure the required pixel clock following the panel datasheet.
- Configure the synchronous timings: VSYNC, HSYNC, vertical and horizontal back porch, active data area and the front porch timings following the panel datasheet as described in the [Section 28.4.1: LTDC global configuration parameters](#).
- Configure the synchronous signals and clock polarity in the LTDC\_GCR register.
- If needed, configure the background color in the LTDC\_BCCR register.
- Configure the needed interrupts in the LTDC\_IER and LTDC\_LIPCR register.
- Configure the layer1/2 parameters by:
  - programming the layer window horizontal and vertical position in the LTDC\_LxWHPCR and LTDC\_WVPCR registers. The layer window must be in the active data area.
  - programming the pixel input format in the LTDC\_LxPFCR register
  - programming the color frame buffer start address in the LTDC\_LxCFBAR register
  - programming the line length and pitch of the color frame buffer in the LTDC\_LxCFBLR register
  - programming the number of lines of the color frame buffer in the LTDC\_LxCFBLNR register
  - if needed, loading the CLUT with the RGB values and its address in the LTDC\_LxCLUTWR register
  - If needed, configuring the default color and the blending factors respectively in the LTDC\_LxDCCR and LTDC\_LxBFCR registers
- Enable layer1/2 and if needed the CLUT in the LTDC\_LxCR register.
- If needed, enable dithering and color keying respectively in the LTDC\_GCR and LTDC\_LxCKCR registers. They can be also enabled on the fly.
- Reload the shadow registers to active register through the LTDC\_SRCR register.
- Enable the LCD-TFT controller in the LTDC\_GCR register.
- All layer parameters can be modified on the fly except the CLUT. The new configuration has to be either reloaded immediately or during vertical blanking period by configuring the LTDC\_SRCR register.

*Note:* All layer's registers are shadowed. Once a register is written, it must not be modified again before the reload has been done. Thus, a new write to the same register overrides the previous configuration if not yet reloaded.

## 28.7 LTDC registers

### 28.7.1 LTDC synchronization size configuration register (LTDC\_SSCR)

This register defines the number of horizontal synchronization pixels minus 1 and the number of vertical synchronization lines minus 1. Refer to [Figure 172](#) and [Section 28.4: LTDC programmable parameters](#) for an example of configuration.

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.	HSW[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	VSH[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 27:16 **HSW[11:0]**: horizontal synchronization width (in units of pixel clock period)  
 These bits define the number of Horizontal Synchronization pixel minus 1.

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

Bits 10:0 **VSH[10:0]**: vertical synchronization height (in units of horizontal scan line)  
 These bits define the vertical Synchronization height minus 1. It represents the number of horizontal synchronization lines.

### 28.7.2 LTDC back porch configuration register (LTDC\_BPCR)

This register defines the accumulated number of horizontal synchronization and back porch pixels minus 1 (**HSYNC width + HBP - 1**) and the accumulated number of vertical synchronization and back porch lines minus 1 (**VSYNC height + VBP - 1**). Refer to [Figure 172](#) and [Section 28.4: LTDC programmable parameters](#) for an example of configuration.

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.	AHBP[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	AVBP[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 27:16 **AHBP[11:0]**: accumulated horizontal back porch (in units of pixel clock period)  
 These bits define the accumulated horizontal back porch width that includes the horizontal synchronization and horizontal back porch pixels minus 1.  
 The horizontal back porch is the period between horizontal synchronization going inactive and the start of the active display part of the next scan line.

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

Bits 10:0 **AVBP[10:0]**: accumulated Vertical back porch (in units of horizontal scan line)  
 These bits define the accumulated vertical back porch width that includes the vertical synchronization and vertical back porch lines minus 1.  
 The vertical back porch is the number of horizontal scan lines at a start of frame to the start of the first active scan line of the next frame.

### 28.7.3 LTDC active width configuration register (LTDC\_AWCR)

This register defines the accumulated number of horizontal synchronization, back porch and active pixels minus 1 (**HSYNC width + HBP + active width - 1**) and the accumulated number of vertical synchronization, back porch lines and active lines minus 1 (**VSYNC height + BVBP + active height - 1**). Refer to [Figure 172](#) and [Section 28.4: LTDC programmable parameters](#) for an example of configuration.

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.	AAW[11:0]											
				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	AAH[10:0]										
					r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 27:16 **AAW[11:0]**: accumulated active width (in units of pixel clock period)  
 These bits define the accumulated active width which includes the horizontal synchronization, horizontal back porch and active pixels minus 1.  
 The active width is the number of pixels in active display area of the panel scan line.  
 Refer to device datasheet for maximum active width supported following maximum pixel clock.

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

Bits 10:0 **AAH[10:0]**: accumulated active height (in units of horizontal scan line)  
 These bits define the accumulated height which includes the vertical synchronization, vertical back porch and the active height lines minus 1. The active height is the number of active lines in the panel.  
 Refer to device datasheet for maximum active height supported following maximum pixel clock.

### 28.7.4 LTDC total width configuration register (LTDC\_TWCR)

This register defines the accumulated number of horizontal synchronization, back porch, active and front porch pixels minus 1 (**HSYNC width + HBP + active width + HFP - 1**) and the accumulated number of vertical synchronization, back porch lines, active and front lines minus 1 (**VSYNC height + BVBP + active height + VFP - 1**). Refer to [Figure 172](#) and [Section 28.4: LTDC programmable parameters](#) for an example of configuration.

Address offset: 0x14

Reset value: 0x0000 0000

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

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

Bits 27:16 **TOTALW[11:0]**: total width (in units of pixel clock period)

These bits defines the accumulated total width which includes the horizontal synchronization, horizontal back porch, active width and horizontal front porch pixels minus 1.

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

Bits 10:0 **TOTALH[10:0]**: total height (in units of horizontal scan line)

These bits defines the accumulated height which includes the vertical synchronization, vertical back porch, the active height and vertical front porch height lines minus 1.

### 28.7.5 LTDC global control register (LTDC\_GCR)

This register defines the global configuration of the LCD-TFT controller.

Address offset: 0x18

Reset value: 0x0000 2220

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HSPOL	VSPOL	DEPOL	PCPOL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEN
rw	rw	rw	rw												rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DRW[2:0]			Res.	DGW[2:0]			Res.	DBW[2:0]			Res.	Res.	Res.	LTCEN
	r	r	r		r	r	r		r	r	r				rw

- Bit 31 **HSPOL**: horizontal synchronization polarity  
This bit is set and cleared by software.  
0: horizontal synchronization polarity is active low.  
1: horizontal synchronization polarity is active high.
- Bit 30 **VSPOL**: vertical synchronization polarity  
This bit is set and cleared by software.  
0: vertical synchronization is active low.  
1: vertical synchronization is active high.
- Bit 29 **DEPOL**: not data enable polarity  
This bit is set and cleared by software.  
0: not data enable polarity is active low.  
1: not data enable polarity is active high.
- Bit 28 **PCPOL**: pixel clock polarity  
This bit is set and cleared by software.  
0: pixel clock polarity is active low.  
1: pixel clock is active high.
- Bits 27:17 Reserved, must be kept at reset value.
- Bit 16 **DEN**: dither enable  
This bit is set and cleared by software.  
0: dither disable  
1: dither enable
- Bit 15 Reserved, must be kept at reset value.
- Bits 14:12 **DRW[2:0]**: dither red width  
These bits return the Dither Red Bits.
- Bit 11 Reserved, must be kept at reset value.
- Bits 10:8 **DGW[2:0]**: dither green width  
These bits return the dither green bits.
- Bit 7 Reserved, must be kept at reset value.
- Bits 6:4 **DBW[2:0]**: dither blue width  
These bits return the dither blue bits.
- Bits 3:1 Reserved, must be kept at reset value.
- Bit 0 **LTDCEN**: LCD-TFT controller enable  
This bit is set and cleared by software.  
0: LTDC disable  
1: LTDC enable

### 28.7.6 LTDC shadow reload configuration register (LTDC\_SRCR)

This register allows to reload either immediately or during the vertical blanking period, the shadow registers values to the active registers. The shadow registers are all Layer1 and Layer2 registers except the LTDC\_L1CLUTWR and the LTDC\_L2CLUTWR.

Address offset: 0x24

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **VBR**: vertical blanking reload

This bit is set by software and cleared only by hardware after reload (it cannot be cleared through register write once it is set).

0: no effect

1: The shadow registers are reloaded during the vertical blanking period (at the beginning of the first line after the active display area).

Bit 0 **IMR**: immediate reload

This bit is set by software and cleared only by hardware after reload.

0: no effect

1: The shadow registers are reloaded immediately.

*Note:* The shadow registers read back the active values. Until the reload has been done, the 'old' value is read.

### 28.7.7 LTDC background color configuration register (LTDC\_BCCR)

This register defines the background color (RGB888).

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.	BCRED[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
BCGREEN[7:0]								BCBLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:16 **BCRED[7:0]**: background color red value  
 These bits configure the background red value.

Bits 15:8 **BCGREEN[7:0]**: background color green value  
 These bits configure the background green value.

Bits 7:0 **BCBLUE[7:0]**: background color blue value  
 These bits configure the background blue value.

### 28.7.8 LTDC interrupt enable register (LTDC\_IER)

This register determines which status flags generate an interrupt request by setting the corresponding bit to 1.

Address offset: 0x34

Reset value: 0x0000 0000

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

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

Bit 3 **RRIE**: register reload interrupt enable  
 This bit is set and cleared by software.  
 0: register reload interrupt disable  
 1: register reload interrupt enable

Bit 2 **TERRIE**: transfer error interrupt enable  
 This bit is set and cleared by software.  
 0: transfer error interrupt disable  
 1: transfer error interrupt enable

Bit 1 **FUIE**: FIFO underrun interrupt enable  
 This bit is set and cleared by software.  
 0: FIFO underrun interrupt disable  
 1: FIFO underrun Interrupt enable

Bit 0 **LIE**: line interrupt enable  
 This bit is set and cleared by software.  
 0: line interrupt disable  
 1: line interrupt enable



### 28.7.9 LTDC interrupt status register (LTDC\_ISR)

This register returns the interrupt status flag.

Address offset: 0x38

Reset value: 0x0000 0000

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

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

Bit 3 **RRIF**: register reload interrupt flag

0: no register reload interrupt generated

1: register reload interrupt generated when a vertical blanking reload occurs (and the first line after the active area is reached)

Bit 2 **TERRIF**: transfer error interrupt flag

0: no transfer error interrupt generated

1: transfer error interrupt generated when a bus error occurs

Bit 1 **FUIF**: FIFO underrun interrupt flag

0: no FIFO underrun interrupt generated.

1: FIFO underrun interrupt generated, if one of the layer FIFOs is empty and pixel data is read from the FIFO

Bit 0 **LIF**: line interrupt flag

0: no line interrupt generated

1: line interrupt generated when a programmed line is reached

### 28.7.10 LTDC Interrupt Clear Register (LTDC\_ICR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRRIF	CTERRIF	CFUIF	CLIF
												w	w	w	w

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

Bit 3 **CRRIF**: clears register reload interrupt flag

- 0: no effect
- 1: clears the RRIF flag in the LTDC\_ISR register

Bit 2 **CTERRIF**: clears the transfer error interrupt flag

- 0: no effect
- 1: clears the TERRIF flag in the LTDC\_ISR register.

Bit 1 **CFUIF**: clears the FIFO underrun interrupt flag

- 0: no effect
- 1: clears the FUDERRIF flag in the LTDC\_ISR register.

Bit 0 **CLIF**: clears the line interrupt flag

- 0: no effect
- 1: clears the LIF flag in the LTDC\_ISR register.

### 28.7.11 LTDC line interrupt position configuration register (LTDC\_LIPCR)

This register defines the position of the line interrupt. The line value to be programmed depends on the timings parameters. Refer to [Figure 172](#).

Address offset: 0x40

Reset value: 0x0000 0000

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

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

Bits 10:0 **LIPOS[10:0]**: line interrupt position

These bits configure the line interrupt position.

### 28.7.12 LTDC current position status register (LTDC\_CPSR)

Address offset: 0x44

Reset value: 0x0000 0000

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

Bits 31:16 **CXPOS[15:0]**: current X position  
 These bits return the current X position.

Bits 15:0 **CYPOS[15:0]**: current Y position  
 These bits return the current Y position.

### 28.7.13 LTDC current display status register (LTDC\_CDSR)

This register returns the status of the current display phase which is controlled by the HSYNC, VSYNC, and horizontal/vertical DE signals.

Example: if the current display phase is the vertical synchronization, the VSYNCS bit is set (active high). If the current display phase is the horizontal synchronization, the HSYNCS bit is active high.

Address offset: 0x48

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	HSYNCS	VSYNCS	HDES	VDES
												r	r	r	r

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

Bit 3 **HSYNCS**: horizontal synchronization display status  
 0: active low  
 1: active high

Bit 2 **VSYNCS**: vertical synchronization display status  
 0: active low  
 1: active high

Bit 1 **HDES**: horizontal data enable display status  
 0: active low  
 1: active high

Bit 0 **VDES**: vertical data enable display status  
 0: active low  
 1: active high

*Note:* The returned status does not depend on the configured polarity in the **LTDC\_GCR** register, instead it returns the current active display phase.

**28.7.14 LTDC layer x control register (LTDC\_LxCR)**

Address offset: 0x84 + 0x80 \* (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	CLUTEN	Res.	Res.	COLKEN	LEN
											rw			rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CLUTEN**: color look-up table enable

This bit is set and cleared by software.

0: color look-up table disable

1: color look-up table enable

The CLUT is only meaningful for L8, AL44 and AL88 pixel format. Refer to [Color look-up table \(CLUT\)](#)

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

Bit 1 **COLKEN**: color keying enable

This bit is set and cleared by software.

0: color keying disable

1: color keying enable

Bit 0 **LEN**: layer enable

This bit is set and cleared by software.

0: layer disable

1: layer enable

### 28.7.15 LTDC layer x window horizontal position configuration register (LTDC\_LxWHPCR)

This register defines the horizontal position (first and last pixel) of the layer 1 or 2 window.

The first visible pixel of a line is the programmed value of AHBP[11:0] bits + 1 in the LTDC\_BPCR register.

The last visible pixel of a line is the programmed value of AAW[10:0] bits in the LTDC\_AWCR register.

Address offset:  $0x88 + 0x80 * (x - 1)$ , ( $x = 1$  to  $2$ )

Reset value: 0x0000 0000

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

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

Bits 27:16 **WHSPPOS[11:0]**: window horizontal stop position

These bits configure the last visible pixel of a line of the layer window.

WHSPPOS[11:0] must be  $\geq$  **AHBP[11:0] bits + 1** (programmed in LTDC\_BPCR register).

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

Bits 11:0 **WHSTPOS[11:0]**: window horizontal start position

These bits configure the first visible pixel of a line of the layer window.

WHSTPOS[11:0] must be  $\leq$  **AAW[11:0] bits** (programmed in LTDC\_AWCR register).

Example:

The LTDC\_BPCR register is configured to 0x000E0005 (AHBP[11:0] is 0xE) and the LTDC\_AWCR register is configured to 0x028E01E5 (AAW[11:0] is 0x28E). To configure the horizontal position of a window size of 630x460, with horizontal start offset of 5 pixels in the active data area:

1. layer window first pixel, WHSTPOS[11:0], must be programmed to 0x14 (0xE+1+0x5)
2. layer window last pixel, WHSPPOS[11:0], must be programmed to 0x28A.

### 28.7.16 LTDC layer x window vertical position configuration register (LTDC\_LxWVPCR)

This register defines the vertical position (first and last line) of the layer1 or 2 window.

The first visible line of a frame is the programmed value of AVBP[10:0] bits + 1 in the register LTDC\_BPCR register.

The last visible line of a frame is the programmed value of AAH[10:0] bits in the LTDC\_AWCR register.

Address offset:  $0x8C + 0x80 * (x - 1)$ , (x = 1 to 2)

Reset value: 0x0000 0000

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

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

Bits 26:16 **WVSPPOS[10:0]**: window vertical stop position

These bits configure the last visible line of the layer window.

WVSPPOS[10:0] must be  $\geq$  **AVBP[10:0] bits + 1** (programmed in LTDC\_BPCR register).

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

Bits 10:0 **WVSTPOS[10:0]**: window vertical start position

These bits configure the first visible line of the layer window.

WVSTPOS[10:0] must be  $\leq$  **AAH[10:0] bits** (programmed in LTDC\_AWCR register).

Example:

The LTDC\_BPCR register is configured to 0x000E0005 (AVBP[10:0] is 0x5) and the LTDC\_AWCR register is configured to 0x028E01E5 (AAH[10:0] is 0x1E5).

To configure the vertical position of a window size of 630x460, with vertical start offset of 8 lines in the active data area:

1. layer window first line: WVSTPOS[10:0] must be programmed to 0xE (0x5 + 1 + 0x8).
2. layer window last line: WVSPPOS[10:0] must be programmed to 0x1DA.

### 28.7.17 LTDC layer x color keying configuration register (LTDC\_LxCKCR)

This register defines the color key value (RGB), that is used by the color keying.

Address offset:  $0x90 + 0x80 * (x - 1)$ , ( $x = 1$  to  $2$ )

Reset value:  $0x0000\ 0000$

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

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

Bits 23:16 **CKRED[7:0]**: color key red value

Bits 15:8 **CKGREEN[7:0]**: color key green value

Bits 7:0 **CKBLUE[7:0]**: color key blue value

### 28.7.18 LTDC layer x pixel format configuration register (LTDC\_LxPFCR)

This register defines the pixel format that is used for the stored data in the frame buffer of a layer. The pixel data is read from the frame buffer and then transformed to the internal format 8888 (ARGB).

Address offset:  $0x94 + 0x80 * (x - 1)$ , ( $x = 1$  to  $2$ )

Reset value:  $0x0000\ 0000$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	PF[2:0]		
													rw	rw	rw

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

Bits 2:0 **PF[2:0]**: pixel format  
 These bits configure the pixel format  
 000: ARGB8888  
 001: RGB888  
 010: RGB565  
 011: ARGB1555  
 100: ARGB4444  
 101: L8 (8-bit luminance)  
 110: AL44 (4-bit alpha, 4-bit luminance)  
 111: AL88 (8-bit alpha, 8-bit luminance)

**28.7.19 LTDC layer x constant alpha configuration register (LTDC\_LxCACR)**

This register defines the constant alpha value (divided by 255 by hardware), that is used in the alpha blending. Refer to LTDC\_LxBFCR register.

Address offset:  $0x98 + 0x80 * (x - 1)$ , (x = 1 to 2)

Reset value: 0x0000 00FF

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

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

Bits 7:0 **CONSTA[7:0]**: constant alpha  
 These bits configure the constant alpha used for blending. The constant alpha is divided by 255 by hardware.  
 Example: if the programmed constant alpha is 0xFF, the constant alpha value is  $255 / 255 = 1$ .



**28.7.20 LTDC layer x default color configuration register (LTDC\_LxDCCR)**

This register defines the default color of a layer in the format ARGB. The default color is used outside the defined layer window or when a layer is disabled. The reset value of 0x00000000 defines a transparent black color.

Address offset:  $0x9C + 0x80 * (x - 1)$ , ( $x = 1$  to  $2$ )

Reset value: 0x0000 0000

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

- Bits 31:24 **DCALPHA[7:0]**: default color alpha  
These bits configure the default alpha value.
- Bits 23:16 **DCRED[7:0]**: default color red  
These bits configure the default red value.
- Bits 15:8 **DCGREEN[7:0]**: default color green  
These bits configure the default green value.
- Bits 7:0 **DCBLUE[7:0]**: default color blue  
These bits configure the default blue value.

### 28.7.21 LTDC layer x blending factors configuration register (LTDC\_LxBFCR)

This register defines the blending factors F1 and F2.

The general blending formula is:  $BC = BF1 \times C + BF2 \times Cs$

- BC = blended color
- BF1 = blend factor 1
- C = current layer color
- BF2 = blend factor 2
- Cs = subjacent layers blended color

Address offset:  $0xA0 + 0x80 \times (x - 1)$ , (x = 1 to 2)

Reset value: 0x0000 0607

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	BF1[2:0]			Res.	Res.	Res.	Res.	Res.	BF2[2:0]		
					rw	rw	rw						rw	rw	rw

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

Bits 10:8 **BF1[2:0]**: blending factor 1

These bits select the blending factor F1.

- 000: reserved
- 001: reserved
- 010: reserved
- 011: reserved
- 100: constant alpha
- 101: reserved
- 110: pixel alpha x constant alpha
- 111: reserved

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

Bits 2:0 **BF2[2:0]**: blending factor 2

These bits select the blending factor F2

- 000: reserved
- 001: reserved
- 010: reserved
- 011: reserved
- 100: reserved
- 101: 1 - constant alpha
- 110: reserved
- 111: 1 - (pixel alpha x constant alpha)

**Note:** The constant alpha value, is the programmed value in the LxCACR register divided by 255 by hardware.

**Example:** Only layer1 is enabled, BF1 configured to constant alpha. BF2 configured to 1 - constant alpha. The constant alpha programmed in the LxCACR register is 240 (0xF0). Thus, the constant alpha value is 240/255 = 0.94. C: current layer color is 128. Cs: background color is 48. Layer1 is blended with the background color.  
 $BC = \text{constant alpha} \times C + (1 - \text{Constant Alpha}) \times Cs = 0.94 \times 128 + (1 - 0.94) \times 48 = 123.$

### 28.7.22 LTDC layer x color frame buffer address register (LTDC\_LxCFBAR)

This register defines the color frame buffer start address which has to point to the address where the pixel data of the top left pixel of a layer is stored in the frame buffer.

Address offset:  $0xAC + 0x80 * (x - 1)$ , (x = 1 to 2)

Reset value: 0x0000 0000

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

Bits 31:0 **CFBADD[31:0]**: color frame buffer start address  
 These bits define the color frame buffer start address.

### 28.7.23 LTDC layer x color frame buffer length register (LTDC\_LxCFBLR)

This register defines the color frame buffer line length and pitch.

Address offset:  $0xB0 + 0x80 * (x - 1)$ , (x = 1 to 2)

Reset value: 0x0000 0000

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **CFBP[12:0]**: color frame buffer pitch in bytes  
 These bits define the pitch that is the increment from the start of one line of pixels to the start of the next line in bytes.

Bits 15:13 Reserved, must be kept at reset value.



Bits 12:0 **CFBLL[12:0]**: color frame buffer line length  
 These bits define the length of one line of pixels in bytes + 3.  
 The line length is computed as follows:  
 active high width \* number of bytes per pixel + 3.

Example:

- A frame buffer having the format RGB565 (2 bytes per pixel) and a width of 256 pixels (total number of bytes per line is 256 \* 2 = 512), where pitch = line length requires a value of 0x02000203 to be written into this register.
- A frame buffer having the format RGB888 (3 bytes per pixel) and a width of 320 pixels (total number of bytes per line is 320 \* 3 = 960), where pitch = line length requires a value of 0x03C003C3 to be written into this register.

**28.7.24 LTDC layer x color frame buffer line number register (LTDC\_LxCFBLNR)**

This register defines the number of lines in the color frame buffer.

Address offset: 0xB4 + 0x80 \* (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	CFBLNBR[10:0]										
					r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 10:0 **CFBLNBR[10:0]**: frame buffer line number  
 These bits define the number of lines in the frame buffer that corresponds to the active high width.

*Note:* The number of lines and line length settings define how much data is fetched per frame for every layer. If it is configured to less bytes than required, a FIFO underrun interrupt will be generated if enabled.  
 The start address and pitch settings on the other hand define the correct start of every line in memory.

**28.7.25 LTDC layer x CLUT write register (LTDC\_LxCLUTWR)**

This register defines the CLUT address and the RGB value.

Address offset:  $0xC4 + 0x80 * (x - 1)$ , (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLUTADD[7:0]								RED[7:0]							
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
GREEN[7:0]								BLUE[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:24 **CLUTADD[7:0]**: CLUT address

These bits configure the CLUT address (color position within the CLUT) of each RGB value.

Bits 23:16 **RED[7:0]**: red value

These bits configure the red value.

Bits 15:8 **GREEN[7:0]**: green value

These bits configure the green value.

Bits 7:0 **BLUE[7:0]**: blue value

These bits configure the blue value.

*Note: The CLUT write register should only be configured during blanking period or if the layer is disabled. The CLUT can be enabled or disabled in the LTDC\_LxCR register.*

*The CLUT is only meaningful for L8, AL44 and AL88 pixel format.*

### 28.7.26 LTDC register map

The following table summarizes the LTDC registers. Refer to the register boundary addresses table for the LTDC register base address.

**Table 174. LTDC 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		
0x0008	LTDC_SSCR	Res.	Res.	Res.	Res.	HSW[11:0]											Res.	Res.	Res.	Res.	Res.	VSH[10:0]													
	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	
0x000C	LTDC_BPCR	Res.	Res.	Res.	Res.	AHBP[11:0]											Res.	Res.	Res.	Res.	Res.	AVBP[10:0]													
	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
0x0010	LTDC_AWCR	Res.	Res.	Res.	Res.	AAW[11:0]											Res.	Res.	Res.	Res.	Res.	AAH[10:0]													
	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
0x0014	LTDC_TWCR	Res.	Res.	Res.	Res.	TOTALW[11:0]											Res.	Res.	Res.	Res.	Res.	TOTALH[10:0]													
	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
0x0018	LTDC_GCR	HSPOL	VSPOL	DEPOL	PCPOL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEN	Res.	DRW[2:0]		Res.	DGW[2:0]		Res.	DBW[2:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTCEN
	Reset value	0	0	0	0													0		0	1	0		0	1	0		0	1	0				0	
0x0024	LTDC_SRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBR	IMR	
	Reset value																															0	0		
0x002C	LTDC_BCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCRED[7:0]							BCGREEN[7:0]							BCBLUE[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
0x0034	LTDC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LIE
	Reset value																																0	0	0
0x0038	LTDC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LIF
	Reset value																																0	0	0
0x003C	LTDC_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.	CLIF
	Reset value																															0	0	0	0
0x0040	LTDC_LIPCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LIPOS
	Reset value																																	0	0
0x0044	LTDC_CPSR	CXPOS[15:0]															CYPOS[15:0]																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Table 174. LTDC 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	LTDC_CDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0x0084	LTDC_L1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0x0088	LTDC_L1WHPCR	Res.	Res.	Res.	Res.	WHSPPPOS[11:0]										WHSTPOS[11:0]																	
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008C	LTDC_L1WVPCR	Res.	Res.	Res.	Res.	Res.	WVSPPOS[10:0]										WVSTPOS[10:0]																
	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
0x0090	LTDC_L1CKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKRED[7:0]							CKGREEN[7:0]							CKBLUE[7:0]									
	Reset value																																
0x0094	LTDC_L1PFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0x0098	LTDC_L1CACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0x009C	LTDC_L1DCCR	DCALPHA[7:0]							DCRED[7:0]							DCGREEN[7:0]							DCBLUE[7:0]										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A0	LTDC_L1BFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0x00AC	LTDC_L1CFBAR	CFBADD[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
0x00B0	LTDC_L1CFBLR	Res.	Res.	Res.	CFBPP[12:0]												CFBLL[12:0]																
	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
0x00B4	LTDC_L1CFBLNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0x00C4	LTDC_L1CLUTWR	CLUTADD[7:0]							RED[7:0]							GREEN[7:0]							BLUE[7:0]										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0104	LTDC_L2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																





## 29 DSI Host (DSIHOST)

### 29.1 Introduction

The Display Serial Interface (DSI) is part of a group of communication protocols defined by the MIPI® Alliance. The MIPI® DSI Host Controller is a digital core that implements all protocol functions defined in the MIPI® DSI Specification.

It provides an interface between the system and the MIPI® D-PHY, allowing the communication with a DSI-compliant display.

### 29.2 Standard and references

- MIPI® Alliance Specification for Display Serial Interface (DSI)  
v1.1 - 22 November 2011
- MIPI® Alliance Specification for Display Bus Interface (DBI-2)  
v2.00 - 16 November 2005
- MIPI® Alliance Specification for Display Command Set (DCS)  
v1.1 - 22 November 2011
- MIPI® Alliance Specification for Display Pixel Interface (DPI-2)  
v2.00 - 15 September 2005
- MIPI® Alliance Specification for Stereoscopic Display Formats (SDF)  
v1.0 - 22 November 2011
- MIPI® Alliance Specification for D-PHY  
v1.1 - 7 November 2011

## 29.3 DSI Host main features

- Compliant with MIPI<sup>®</sup> Alliance standards (see [Section 29.2: Standard and references](#))
- Interface with MIPI<sup>®</sup> D-PHY
- Supports all commands defined in the MIPI<sup>®</sup> Alliance specification for DCS:
  - Transmission of all Command mode packets through the APB interface
  - Transmission of commands in Low-Power and High-Speed during Video mode
- Supports up to two D-PHY data lanes
- Bidirectional communication and escape mode support through data lane 0
- Supports non-continuous clock in D-PHY clock lane for additional power saving
- Supports Ultra Low-Power mode with PLL disabled
- ECC and Checksum capabilities
- Support for End of Transmission Packet (EoTp)
- Fault recovery schemes
- Configurable selection of system interfaces:
  - AMBA APB for control and optional support for Generic and DCS commands
  - Video mode interface through LTDC
  - Adapted Command mode interface through LTDC
  - Independently programmable virtual channel Id in Video mode, Adapted Command mode and APB slave
- Video mode interfaces features:
  - LTDC interface color coding mappings into 24-bit interface:
    - 16-bit RGB, configurations 1, 2, and 3
    - 18-bit RGB, configurations 1 and 2
    - 24-bit RGB
  - Programmable polarity of all LTDC interface signals
  - Extended resolutions beyond the DPI standard maximum resolution of 800x480 pixels:
  - Maximum resolution is limited by available DSI physical link bandwidth:
    - Number of lanes: 2
    - Maximum speed per lane: 500 Mbps
    - See examples in [Section 29.4.2: Supported resolutions and frame rates](#)
- Adapted interface features:
  - Support for sending large amounts of data through the *memory\_write\_start* (WMS) and *memory\_write\_continue* (WMC) DCS commands
  - LTDC interface color coding mappings into 24-bit interface:
    - 16-bit RGB, configurations 1, 2, and 3
    - 18-bit RGB, configurations 1 and 2
    - 24-bit RGB
- Video mode pattern generator:
  - Vertical and horizontal color bar generation without LTDC stimuli
  - BER pattern without LTDC stimuli

## 29.4 DSI Host functional description

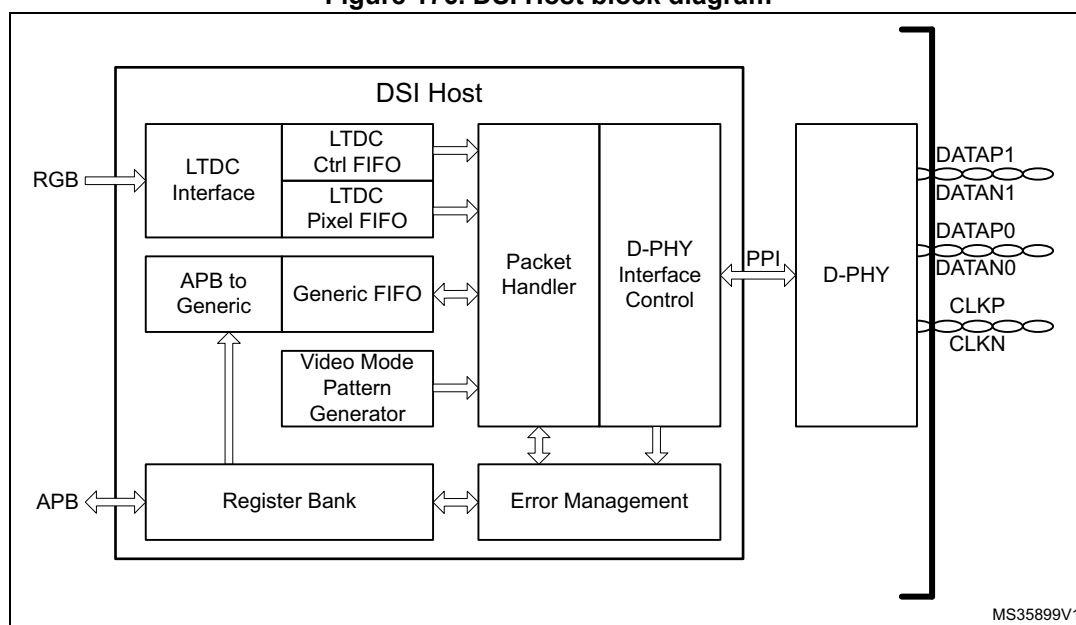
### 29.4.1 General description

The MIPI® DSI Host includes dedicated video interfaces internally connected to the LTDC and a generic APB interface that can be used to transmit information to the display. More in detail:

- LTDC interface:
  - Used to transmit information in Video mode, in which the transfers from the host processor to the peripheral take the form of a real-time pixel stream (DPI).
  - Through a customized mode, this interface can be used to transmit information in full bandwidth in the Adapted Command mode (DBI).
- APB slave interface: This interface allows the transmission of generic information in Command mode, and follows a proprietary register interface. This interface can operate concurrently with either LTDC interface in either Video mode or Adapted Command mode.
- Video mode pattern generator: This interface allows the transmission of horizontal/vertical color bar and D-PHY BER testing pattern without any kind of stimuli.

The block diagram of the DSI Host is shown in [Figure 176](#).

**Figure 176. DSI Host block diagram**



### 29.4.2 Supported resolutions and frame rates

The DSI specification does not define supported standard resolutions or frame rates. Display resolution, blanking periods, synchronization events duration, frame rates, and pixel color depth play a fundamental role in the required bandwidth. In addition, other link related attributes can influence the ability of the link to support a DSI-specific device. These attributes can be: display input buffering capabilities, video transmission mode (Burst or Non-Burst), Bus Turn-Around (BTA) time, concurrent command mode traffic in a video mode transmission, or display device specifics. All these variables make it difficult to define a

standard procedure to estimate the minimum lane rate and the minimum number of lanes that support a specific display device.

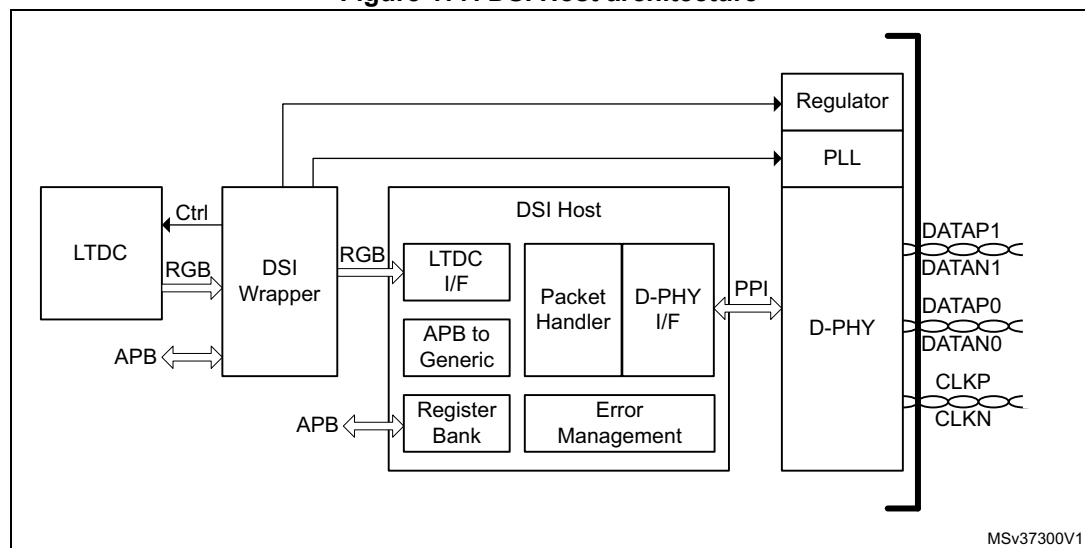
The basic assumptions for estimates are:

- clock lane frequency is 250 MHz, resulting in a bandwidth of 500 Mbps for each data lane;
- the display should be capable of buffering the pixel data at the speed at which it is delivered in the DSI link;
- no significant control traffic is present on the link when the pixel data is being transmitted.

### 29.4.3 System level architecture

Figure 177 shows the architecture of the DSI Host

Figure 177. DSI Host architecture



The different parts have the following functions:

- The DSI Wrapper ensures the interfacing between the LTDC and the DSI Host kernel. It can adapt the color mode, the signal polarity and manages the Tearing Effect (TE) management for automatic frame buffer update in Adapted Command mode. The DSI Wrapper also control the DSI Regulator, the DSI PLL and specific functions of the MIPI<sup>®</sup> D-PHY.
- The LTDC interface captures the data and control signals from the LTDC and conveys them to a FIFO for video control signals and another one for the pixel data. This data is then used to build one of the following:
  - Video packets, when in Video mode (see [Section 29.5](#))
  - The *memory\_write\_start* and *memory\_write\_continue* DCS commands, when in Adapted Command mode (see [Section 29.6](#))
- The Register Bank is accessible through a standard AMBA-APB slave interface, providing access to the DSI Host registers for configuration and control. There is also a fully programmable interrupt generator to inform the system about certain events.
- The PHY Interface Control is responsible for managing the D-PHY interface. It acknowledges the current operation and enables Low-Power transmission/reception or

a High-Speed transmission. It also performs data splitting between available D-PHY lanes for High-Speed transmission.

- The Packet Handler schedules the activities inside the link. It performs several functions based on the interfaces that are currently operational and the video transmission mode that is used (burst mode or non-burst mode with sync pulses or sync events). It builds long or short packet generating correspondent ECC and CRC codes. This block also performs the following functions:
  - packet reception
  - validation of packet header by checking the ECC
  - header correction and notification for single-bit errors
  - termination of reception
  - multiple header error notification
  - depending on the virtual channel of the incoming packet, the handler routes the output data to the respective port.
- The APB-to-Generic block bridges the APB operations into FIFOs holding the Generic commands. The block interfaces with the following FIFOs:
  - Command FIFO
  - Write payload FIFO
  - Read payload FIFO
- The Error Management notifies and monitors the error conditions on the DSI link. It controls the timers used to determine if a timeout condition occurred, performing an internal soft reset and triggering an interruption notification.

## 29.5 Functional description: Video mode on LTDC interface

The LTDC interface captures the data and control signals and conveys them to the FIFO interfaces that transmit them to the DSI link.

Two different streams of data are present at the interface, namely video control signals and pixel data. Depending on the interface color coding, the pixel data is disposed differently throughout the LTDC bus.

Interface pixel color coding is summarized in [Table 175](#).

**Table 175. Location of color components in the LTDC interface**

Location	16 bits			18 bits		24 bits
	Configuration 1	Configuration 2	Configuration 3	Configuration 1	Configuration 2	
D23	-	-	-	-	-	R[7]
D22	-	-	-	-	-	R[6]
D21	-	-	R[4]	-	R[5]	R[5]
D20	-	R[4]	R[3]	-	R[4]	R[4]
D19	-	R[3]	R[2]	-	R[3]	R[3]
D18	-	R[2]	R[1]	-	R[2]	R[2]
D17	-	R[1]	R[0]	R[5]	R[1]	R[1]
D16	-	R[0]	-	R[4]	R[0]	R[0]
D15	R[4]	-	-	R[3]	-	G[7]
D14	R[3]	-	-	R[2]	-	G[6]
D13	R[2]	G[5]	G[5]	R[1]	G[5]	G[5]
D12	R[1]	G[4]	G[4]	R[0]	G[4]	G[4]
D11	R[0]	G[3]	G[3]	G[5]	G[3]	G[3]
D10	G[5]	G[2]	G[2]	G[4]	G[2]	G[2]
D9	G[4]	G[1]	G[1]	G[3]	G[1]	G[1]
D8	G[3]	G[0]	G[0]	G[2]	G[0]	G[0]
D7	G[2]	-	-	G[1]	-	B[7]
D6	G[1]	-	-	G[0]	-	B[6]
D5	G[0]	-	B[4]	B[5]	B[5]	B[5]
D4	B[4]	B[4]	B[3]	B[4]	B[4]	B[4]
D3	B[3]	B[3]	B[2]	B[3]	B[3]	B[3]
D2	B[2]	B[2]	B[1]	B[2]	B[2]	B[2]
D1	B[1]	B[1]	B[0]	B[1]	B[1]	B[1]
D0	B[0]	B[0]	-	B[0]	B[0]	B[0]

The LTDC interface can be configured to increase flexibility and promote correct use of this interface for several systems. The following configuration options are available:

- Polarity control: All the control signals are programmable to change the polarity depending on the LTDC configuration.
- After the core reset, DSI Host waits for the first VSYNC active transition to start signal sampling, including pixel data, thus avoiding starting the transmission of the image data in the middle of a frame.
- If interface pixel color coding is 18 bits and the 18-bit loosely packed stream is disabled, the number of pixels programmed in the VPSIZE field must be a multiple of four. This means that in this mode, the two LSBs in the configuration are always inferred as zero. The specification states that in this mode, the pixel line size should be a multiple of four.
- To avoid FIFO underflows and overflows, the configured number of pixels is assumed to be received from the LTDC at all times.
- To keep the memory organized with respect to the packet scheduling, the number of pixels per packet parameter is used to separate the memory space of different video packets.

For SHTDN and COLM sampling and transmission, the video streaming from the LTDC must be active. This means that if the LTDC is not actively generating the video signals like VSYNC and HSYNC, these signals are not transmitted through the DSI link. Because of such constraints and for commands to be correctly transmitted, the first VSYNC active pulse should occur for the command sampling and transmission. When shutting down the display, it is necessary for the LTDC to be kept active for one frame after the command being issued. This ensures that the commands are correctly transmitted before actually disabling the video generation at the LTDC interface.

The SHTDN and COLM values can be programmed in the DSI Wrapper Control Register (DSI\_WCR).

For all of the data types, one entire pixel is received per each clock cycle. The number of pixels of payload is restricted to a multiple of a value, as shown in [Table 176](#).

**Table 176. Multiplicity of the payload size in pixels for each data type**

Value	Data Types
1	16-bit 18-bit loosely packed 24-bit
2	Loosely packed pixel stream
4	18-bit non-loosely packed

### 29.5.1 Video transmission mode

There are different video transmission modes, namely:

- Burst mode
- Non-Burst mode
  - Non-Burst mode with sync pulse
  - Non-Burst mode with sync event.

### Burst mode

In this mode, the entire active pixel line is buffered into a FIFO and transmitted in a single packet with no interruptions. This transmission mode requires that the DPI Pixel FIFO has the capacity to store a full line of active pixel data inside it. This mode is optimally used when the difference between the pixel required bandwidth and DSI link bandwidth is significant, it enables the DSI Host to quickly dispatch the entire active video line in a single burst of data and then return to Low-power mode.

### Non-Burst mode

In this mode, the processor uses the partitioning properties of the DSI Host to divide the video line transmission into several DSI packets. This is done to match the pixel required bandwidth with the DSI link bandwidth. With this mode, the controller configuration does not require a full line of pixel data to be stored inside the LTDC interface pixel FIFO. It requires only the content of one video packet.

### Guidelines for selecting the Burst or Non-Burst mode

Selecting the Burst and Non-Burst mode is mainly dependent on the system configuration and the device requirements. Choose the video transmission mode that suits the application scenario. The Burst mode is more beneficial because it increases the probability of the link spending more time in the Low-Power mode, decreasing power consumption. However, the following conditions should be met for availing the maximum benefits from the Burst mode of operation:

- The DSI Host core should have sufficient pixel memory to store an entire pixel line to avoid the overflow of the internal FIFOs.
- The display device should support receiving a full pixel line in a single packet burst to avoid the overflow on the reception buffer.
- The DSI output bandwidth should be higher than the LTDC interface input bandwidth in a relation that enables the link to go to Low-Power once per line.

If the system cannot meet these requirements, it is likely that the pixel data will be lost causing the malfunctioning of the display device while using the Burst mode. These errors are related to the capabilities of the system to store the temporary pixel data.

If all the conditions for using the Burst mode cannot be met, use the Non-Burst mode to avoid the errors caused by the Burst mode. The Non-Burst mode provides a better matching of rates for pixel transmission, enabling:

- Only a certain amount of pixels to be stored in the memory and not requiring a full pixel line (lesser LTDC interface RAM requirements in the DSI Host).
- Operation with devices that support only a small amount of pixel buffering (less than a full pixel line).

The DSI Non-Burst mode should be configured in such way that the DSI output pixel ratio matches with the LTDC interface input pixel ratio, reducing the memory requirements on both host and/or device side. This is achieved by dividing a pixel line into several chunks of pixels and optionally interleaving them with null packets.

The following equations show how the DSI Host core transmission parameters should be programmed in Non-Burst mode to match the DSI link pixel output ratio (left hand side of the "=" sign) and LTDC interface pixel input (right hand side of the "=" sign).



When the null packets are enabled:

$$\text{lanebyteclkperiod} * \text{NUMC} (\text{VPSIZE} * \text{bytes\_per\_pixel} + 12 + \text{NPSIZE}) / \text{number\_of\_lanes} \\ = \text{pixels\_per\_line} * \text{LTDC\_Clock\_period}$$

When the null packets are disabled:

$$\text{lanebyteclkperiod} * \text{NUMC} (\text{VPSIZE} * \text{bytes\_per\_pixel} + 6) / \text{number\_of\_lanes} \\ = \text{pixels\_per\_line} * \text{LTDC\_Clock\_period}$$

### 29.5.2 Updating the LTDC interface configuration in video mode

It is possible to update the LTDC interface configuration on the fly without impacting the current frame. It is done with the help of shadow registers. This feature is controlled by the DSI Host Video Shadow Control Register (DSI\_VSCR).

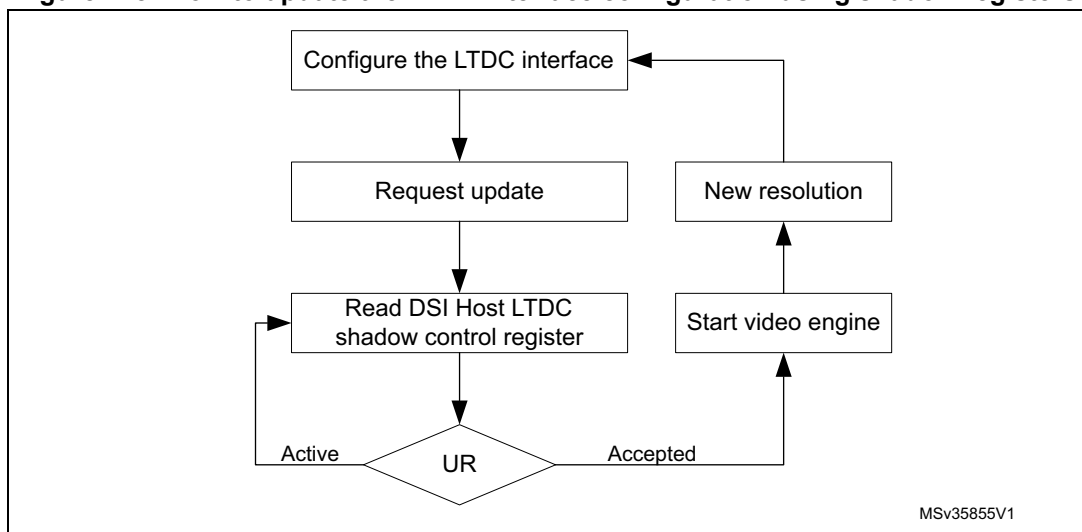
The new configuration is only used when the system requests for it. To update the Video configuration during the transmission of a video frame, the configuration of that frame needs to be stored in the auxiliary registers. This way, the new frame configurations can be set through the APB interface without corrupting the current frame.

By default, this feature is disabled. To enable this feature, set the Enable (EN) bit of the DSI Host Video Shadow Control Register (DSI\_VSCR) to 1.

When this feature is enabled, the system supplies the configuration stored in the auxiliary registers.

Figure 178 shows the necessary steps to update the LTDC interface configuration.

**Figure 178. Flow to update the LTDC interface configuration using shadow registers**

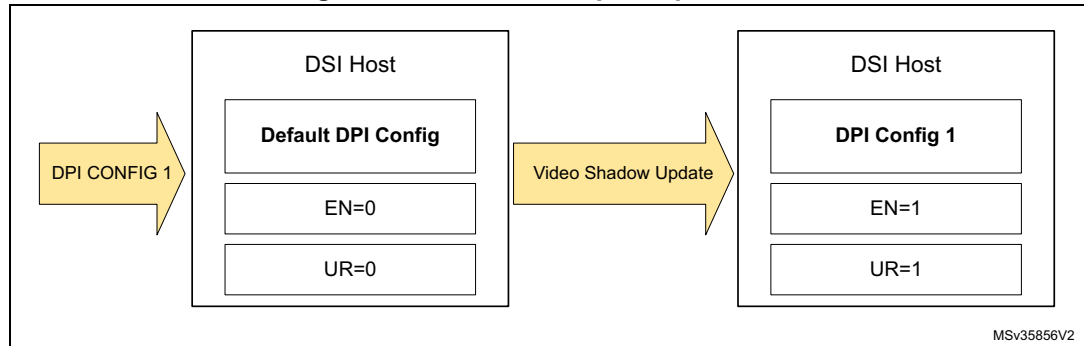


#### Immediate update

When the shadow register feature is active, the auxiliary registers requires the LTDC configuration before the video engine starts. This means that, after a reset, Update Register (UR) bit is immediately granted.

In situations when it is required to immediately update the active registers without the reset (as illustrated in *Figure 179*), ensure that the Enable (EN) and Update Register (UR) bits of the DSI Host Video Shadow Control Register (DSI\_VSCR) are set to 0.

**Figure 179. Immediate update procedure**

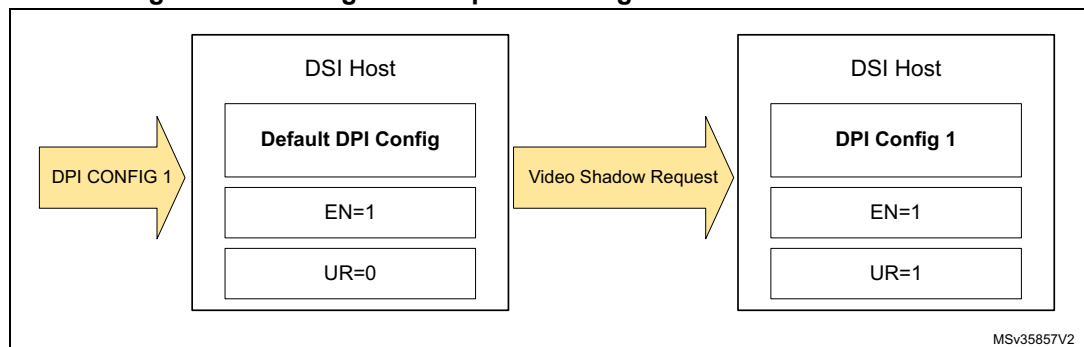


**Updating the configuration during the transmission of a frame using APB**

To update the LTDC interface configuration, follow the steps shown in *Figure 180*:

1. Ensure that the Enable (EN) bit of the DSI Host Video Shadow Control Register (DSI\_VSCR) register is set to 1.
2. Set the Update Register (UR) bit of DSI Host Video Shadow Control Register (DSI\_VSCR) to 1.
3. Monitor the Update Register (UR) bit. This bit is set to 0 when the update is complete.

**Figure 180. Configuration update during the transmission of a frame**



**Requesting a configuration update**

It is possible to request for the LTDC interface configuration update at any part of the frame. DSI Host waits until the end of the frame to change the configuration. However, avoid sending the update request during the first line of the frame because the data must propagate between clock domains.

## 29.6 Functional description: adapted command mode on LTDC interface

The Adapted Command mode, enables the system to input a stream of pixel from the LTDC that is conveyed by DSI Host using the Command mode transmission (using the DCS packets). The Adapted Command mode also supports pixel input control rate signaling and Tearing Effect report mechanism.

The Adapted Command mode allows to send large amounts of data through the *memory\_write\_start* (WMS) and *memory\_write\_continue* (WMC) DCS commands. It helps in delivering a wider data bandwidth for the memory write operations sent in Command mode to MIPI® displays and to refresh large areas of pixels in high resolution displays. If additional commands such as display configuration commands, read back commands, and tearing effect initialization are to be transferred, then the APB slave generic interface should be used to complement the Adapted Command mode functionality.

Adapted Command mode of operation supports 16 bpp, 18 bpp, and 24 bpp RGB.

To transmit the image data in Adapted Command mode:

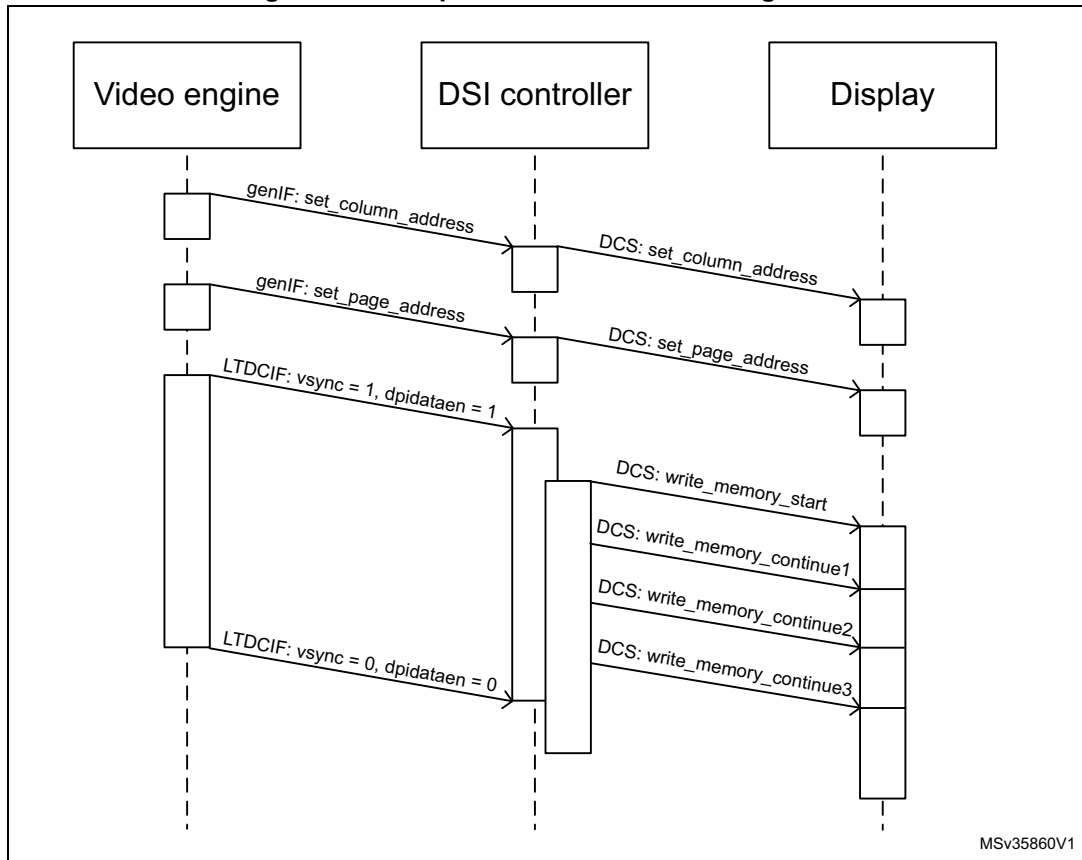
- Set Command mode (CMDM) bit of the DSI Host mode Configuration Register (DSI\_MCR) to 1.
- Set DSI mode (DSIM) bit in the DSI Wrapper Configuration Register (DSI\_WCFGR) to 1.

To transmit the image data, follow these steps:

- Define the image area to be refreshed, by using the *set\_column\_address* and *set\_page\_address* DCS commands. The image area needs to be defined only once and remains effective until different values are defined.
- Define the pixel color coding to be used by using the Color Coding (COLC) field in the DSI Host LTDC Color Coding Register (DSI\_LCOLCR).
- Define the Virtual Channel ID of the LTDC interface generated packets using the Virtual Channel ID (VCID) field in the DSI Host LTDC VCID Register (DSI\_LVCIDR). These also need to be defined only once.
- Start transmitting the data from the LTDC setting the LTDC Enable (LTDCEN) bit of the DSI\_WCR register.

[Figure 181](#) shows the adapted command mode usage flow.

Figure 181. Adapted command mode usage flow



MSv35860V1

When the Command mode (CMDM) bit of the DSI Host mode Configuration Register (DSI\_CFGR) is set to 1, the LTDC interface assume the behavior corresponding to the Adapted Command mode.

In this mode, the host processor can use the LTDC interface to transmit a continuous stream of pixels to be written in the local frame buffer of the peripheral. It uses a pixel input bus to receive the pixels and controls the flow automatically to limit the stream of continuous pixels. When the first pixel is received, the current value of the Command Size (CMDSIZE) field of the DSI Host LTDC Command Configuration Register (DSI\_LCCR), is shadowed to the internal interface function. The interface increments a counter on every valid pixel that is input through the interface. When this pixel counter reaches Command Size (CMDSIZE), a command is written into the command FIFO and the packet is ready to be transmitted through the DSI link.

If the last pixel arrives before the counter reaches the value of shadowed Command Size (CMDSIZE), a WMS command is issued to the command FIFO with Word Count (WC) set to the amount of bytes that correspond to the value of the counter. If more than CMDSIZE number of pixels are received (shadowed value), a WMS command is sent to the command FIFO with WC set to the number of bytes that correspond to Command Size (CMDSIZE) and the counter is restarted.

After the first WMS command has been written to the FIFO, the circuit behaves in a similar way, but issues WMC commands instead of WMS commands. The process is repeated until the last pixel of the image is received. The core automatically starts sending a new packet

when the last pixel of the image is received falls or Command Size (CMDSIZE) limit is reached.

### Synchronization with the LTDC

The DSI wrapper performs the synchronization of the transfer process by :

- controlling the start/halt of the LTDC.
- making the data flow control between LTDC and DSI Host.

The transfer to refresh the display frame buffer can be triggered

- manually, setting the LTDC Enable (LTDCEN) bit of the DSI Wrapper Control Register (DSI\_WCR).
- automatically when a Tearing Effect (TEIF) event occurs and Automatic Refresh (AR) is enabled.

The selection between manual and automatic mode is done through the Automatic Refresh (AR) bit of the DSI Wrapper Configuration Register (DSI\_WCFGR). In automatic refresh mode, the LTDC Enable (LTDCEN) bit of the DSI Wrapper Control Register (DSI\_WCR) is set automatically by a Tearing Effect (TEIF) event.

Once the transfer of one frame is done whatever in manual or automatic refresh mode, the DSI Wrapper is halting the TFT Display Controller (LTDC) resetting the LTDC Enable (LTDCEN) bit of the DSI Wrapper Control Register (DSI\_WCR) and set the End of Refresh Interrupt Flag (ERIF) flag of the DSI Wrapper Status Register (DSI\_WSR). If the End of Refresh Interrupt Enable (ERIE) bit of the DSI Wrapper Configuration Register (DSI\_WCFGR) is set, an interrupt is generated.

The End of Refresh Interrupt Flag (ERIF) flag of the DSI Wrapper Status Register (DSI\_WSR) can be reset setting the Clear End of Refresh Interrupt Flag (CERIF) bit of the DSI Wrapper Clear Interrupt Flag Register (DSI\_WCIFR).

The halting of the TFT Display Controller (LTDC) by the DSI Wrapper is done synchronously on a rising edge or a falling edge of VSync according to the VSync Polarity (VSPOL) bit of the DSI Wrapper Configuration Register (DSI\_WCFGR).

### Support of tearing effect

The DSI specification supports tearing effect function in Command mode displays. It enables the Host Processor to receive timing accurate information about where the display peripheral is in the process of reading the content of its frame buffer.

The Tearing effect can be managed through

- a separate pin which is not covered in the DSI specification
- the DSI tearing effect functionality: a *set\_tear\_on* DCS command should be issued through the APB interface using the Generic interface registers.

### Tearing effect through a GPIO

When the Tearing Effect Source (TESRC) bit of the DSI Wrapper Configuration Register (DSI\_WCFGR) is set, the Tearing effect is signaled through a GPIO.

The polarity of the input signal can be configured by the Tearing Effect Polarity (TEPOL) bit of the DSI Wrapper Configuration Register (DSI\_WCFGR).

When the programmed edge is detected, the Tearing Effect Interrupt Flag (TEIF) bit of the DSI Wrapper Interrupt and Status Register (DSI\_WISR) is set.

If the Tearing Effect Interrupt Enable (TEIE) bit of the DSI Wrapper Interrupt Enable Register (DSI\_WIER) is set, an interrupt is generated.

### Tearing effect through DSI link

When the TESRC bit of the DSI Wrapper Configuration Register (DSI\_WCFGR) is reset, the Tearing effect is managed through the DSI link:

The DSI Host performs a double Bus-Turn-Around (BTA) after sending the *set\_tear\_on* command granting the ownership of the link to the DSI display. The Display holds the ownership of the bus until the tear event occurs, which is indicated to the DSI Host by a D-PHY trigger event. The DSI Host then decodes the trigger and reports the event setting the Tearing Effect Interrupt Flag (TEIF) bit of the DSI Wrapper Interrupt and Status Register (DSI\_WISR).

If the Tearing Effect Interrupt Enable (TEIE) bit of the DSI Wrapper Interrupt Enable Register (DSI\_WIER) is set, an interrupt is generated.

To use this function, it is necessary to issue a *set\_tear\_on* command after the update of the display using the WMS and WMC DCS commands. This procedure halts the DSI link until the display is ready to receive a new frame update.

The DSI Host does not automatically generate the tearing effect request (double BTA) after a WMS/WMC sequence for flexibility purposes. This way several regions of the display can be updated improving DSI bandwidth usage. Tearing effect request must always be triggered by a *set\_tear\_on* command in the DSI Host implementation.

Configure the following registers to activate the tearing effect:

- DSI Host Command mode Configuration Register (DSI\_CMCR): TEARE;
- DSI Host Protocol Configuration Register (DSI\_PCR): BTAE.

## 29.7 Functional description: APB slave generic interface

The APB slave interface allows the transmission of generic information in Command mode, and follows a proprietary register interface. Commands sent through this interface are not constrained to comply with the DCS specification, and can include generic commands described in the DSI specification as manufacturer-specific.

The DSI Host supports the transmission of write and read Command mode packets as described in the DSI specification. These packets are built using the APB register access. The DSI Host Generic Payload Data Register (DSI\_GPDR) has two distinct functions based on the operation. Writing to this register sends the data as payload when sending a Command mode packet. Reading this register returns the payload of a read back operation. The DSI Host Generic Header Configuration Register (DSI\_GHCR) contains the Command mode packet header type and header data. Writing to this register triggers the transmission of the packet implying that for a long Command mode packet, the packet's payload needs to be written in advance in the DSI Host Generic Payload Data Register (DSI\_GPDR).

The valid packets that can be transmitted through the Generic interface are the following ones:

- Generic Write Short Packet 0 Parameters
- Generic Write Short Packet 1 Parameters
- Generic Write Short Packet 2 Parameters
- Generic Read Short Packet 0 Parameters
- Generic Read Short Packet 1 Parameters
- Generic Read Short Packet 2 Parameters
- Maximum Read Packet Configuration
- Generic Long Write Packet
- DCS Write Short Packet 0 Parameters
- DCS Write Short Packet 1 Parameters
- DCS Read Short Packet 0 Parameters
- DCS Write Long Packet.

A set of bits in the DSI Host Generic Packet Status Register (DSI\_GPSR) reports the status of the FIFO associated with APB interface support.

Generic interface packets are always transported using one of the DSI transmission modes, i.e. Video mode or Command mode. If neither of these modes is selected, the packets are not transmitted through the link and the related FIFO eventually becomes overflown.

### 29.7.1 Packet transmission using the generic interface

The transfer of packets through the APB bus is based on the following conditions:

- The APB protocol defines that the write and read procedure takes two clock cycles each to be executed. This means that the maximum input data rate through the APB interface is always half the speed of the APB clock.
- The data input bus has a maximum width of 32 bits. This allows for a relation to be defined between the input APB clock frequency and the maximum bit rate achievable by the APB interface.
- The DSI link pixel bit rate when using solely APB is  $(\text{APB clock frequency}) * 16 \text{ Mbps}$ .
- When using only the APB interface, the theoretical DSI link maximum bit rate can be expressed as  $\text{DSI link maximum bit rate} = \text{APB clock frequency (in MHz)} * 32 / 2 \text{ Mbps}$ . In this formula, the number 32 represents the APB data bus width, and the division by two is present because each APB write procedure takes two clock cycles to be executed.
- The bandwidth is dependent on the APB clock frequency; the available bandwidth increases with the clock frequency.

To drive the APB interface to achieve high bandwidth Command mode traffic transported by the DSI link, the DSI Host should operate in the Command mode only and the APB interface should be the only data source that is currently in use. Thus, the APB interface has the entire bandwidth of the DSI link and does not share it with any another input interface source.

The memory write commands require maximum throughput from the APB interface, because they contain the most amount of data conveyed by the DSI link. While writing the packet information, first write the payload of a given packet into the payload FIFO using the DSI Host Generic Payload Data Register (DSI\_GPDR). When the payload data is for the command parameters, place the first byte to be transmitted in the least significant byte position of the APB data bus.

After writing the payload, write the packet header into the command FIFO. For more information about the packet header organization on the 32-bit APB data bus, so that it is correctly stored inside the Command FIFO.

When the payload data is for a memory write command, it contains pixel information and it should follow the pixel to byte conversion organization referred in the Annexe A of the DCS specification.

Figures 182 to 186 show how the pixel data should be organized in the APB data write bus.

The memory write commands are conveyed in DCS long packets, encapsulated in a DSI packet. The DSI specifies that the DCS command should be present in the first payload byte of the packet. This is also included in the diagrams. In figures 182 to 186, the *Write Memory Command* can be replaced by the DCS command *Write Memory Start* and *Write Memory Continue*.



Figure 182. 24 bpp APB pixel to byte organization

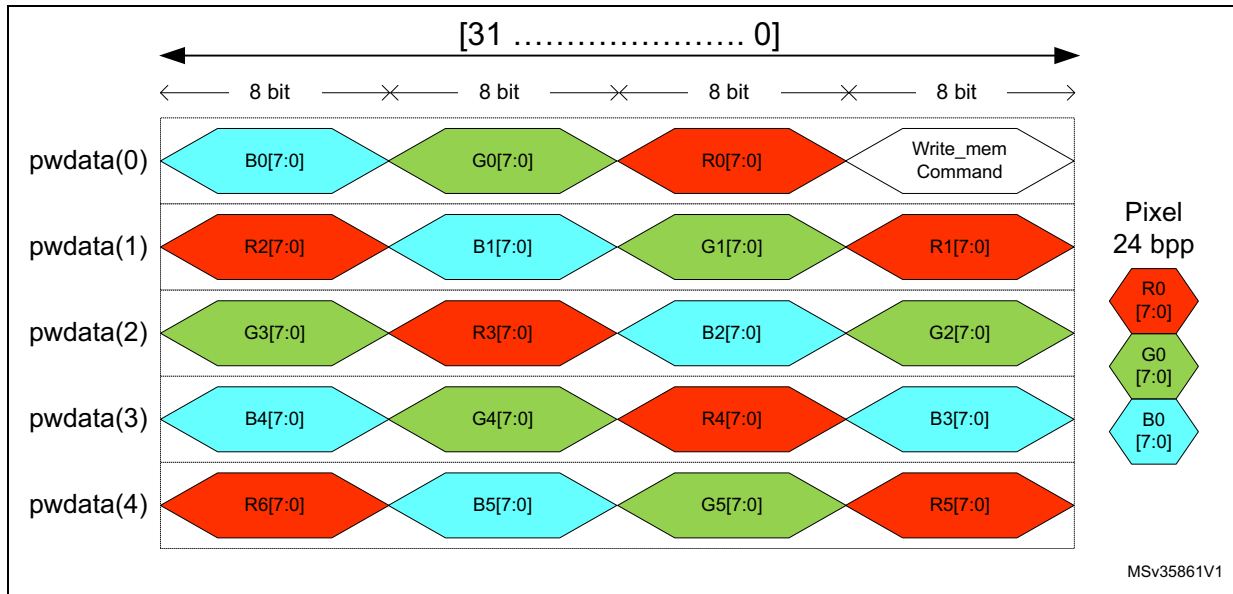


Figure 183. 18 bpp APB pixel to byte organization

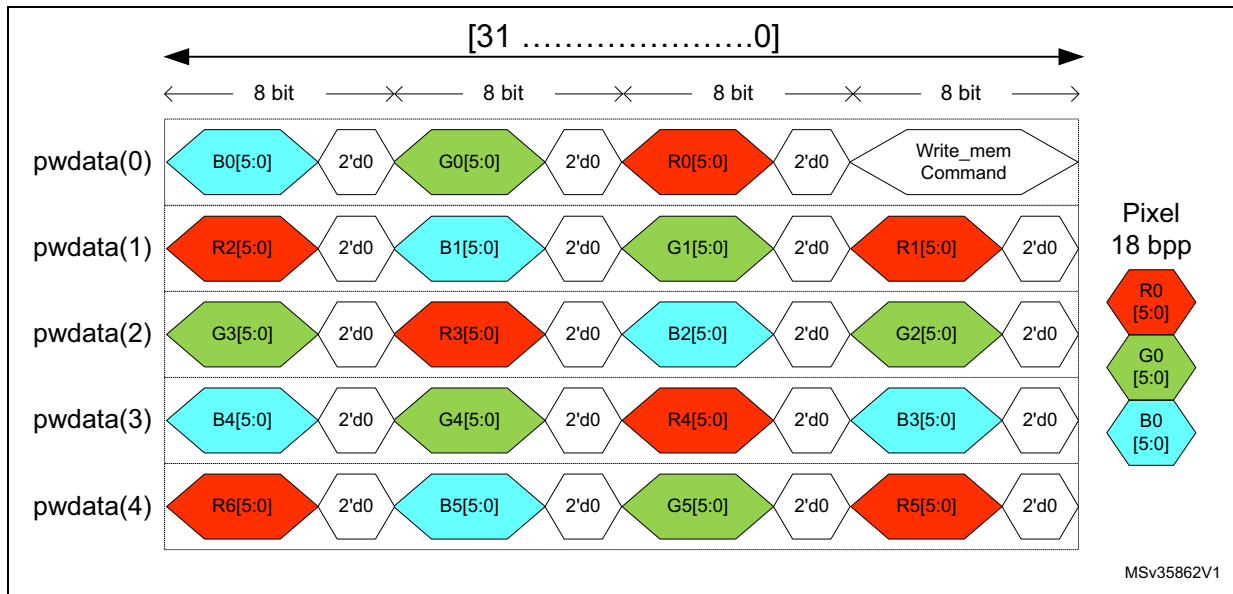


Figure 184. 16 bpp APB pixel to byte organization

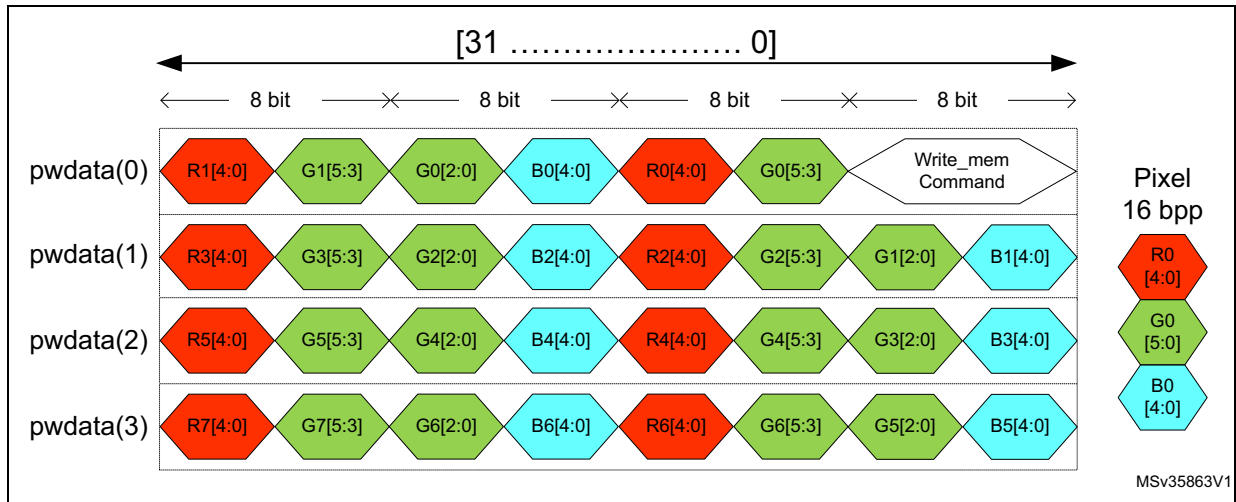


Figure 185. 12 bpp APB pixel to byte organization

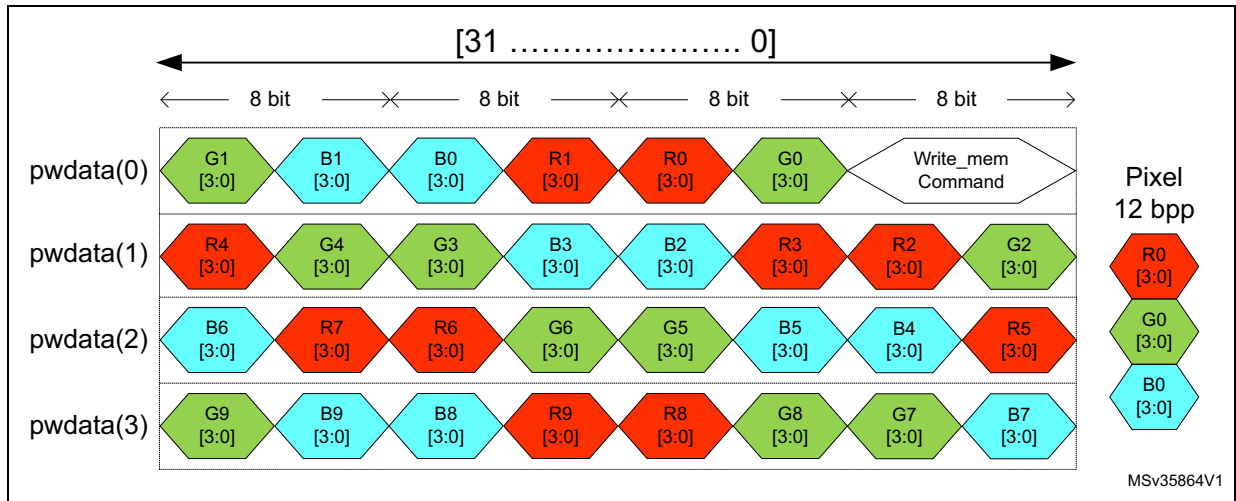
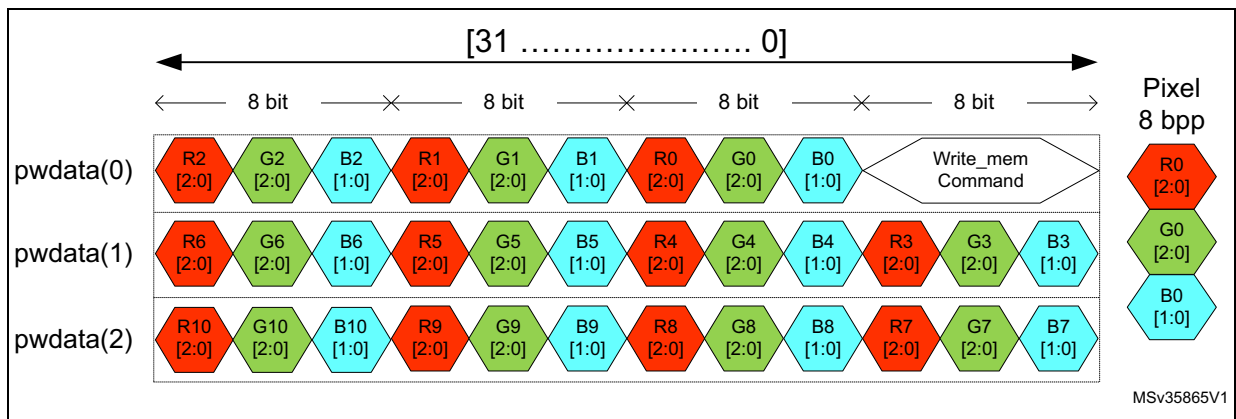


Figure 186. 8 bpp APB pixel to byte organization



## 29.8 Functional description: Timeout counters

The DSI Host includes counters to manage timeout during the various communication phases. The duration of each timeout can be configured by the 6 DSI Host Timeout Counter Configuration Register (DSI\_TCCR0..5).

There are two types of counters:

- contention error detection timeout counters ([Section 29.8.1](#));
- peripheral response timeout counters ([Section 29.8.2](#)).

### 29.8.1 Contention error detection timeout counters

The DSI Host implements a set of counters and conditions to notify the errors. It features a set of registers to control the timers used to determine if a timeout has occurred, and also contains a set of interruption status registers that are cleared upon a read operation (detailed in [Table 177](#)). Optionally, these registers also trigger an interrupt signal that can be used by the system to be activated when an error occurs within the DSI connection.

**Table 177. Contention detection timeout counters configuration**

Timeout counter	Value Register	Value Field	Flag Register	Flag Field
High-speed transmission	DSI_TCCR0	TOHSTX	DSI_ISR1	TOHSTX
Low-power reception	DSI_TCCR0	TOLPRX	DSI_ISR1	TOLPRX

Time units for these 16-bit counters are configured in cycles defined in the Timeout Clock Division (TOCKDIV) field in the DSI Host Clock Control Register (DSI\_CCR).

The value written to the Timeout Clock Division (TOCKDIV) field in the DSI Host Clock Control Register (DSI\_CCR) defines the time unit for the timeout limits using the Lane byte clock as input.

This mechanism increases the range to define these limits.

#### High-speed transmission contention detection

The timeout duration is configured in the High-Speed Transmission Timeout Count (HSTX\_TOCNT) field of the DSI Host Timeout Counter Configuration Register 1 (DSI\_TCCR0). A 16-bit counter measures the time during which the High-Speed mode is active.

If that counter reaches the value defined by the High-Speed Transmission Timeout Count (HSTX\_TOCNT) field of the DSI Host Timeout Counter Configuration Register 1 (DSI\_TCCR0), the Timeout High-Speed Transmission (TOHSTX) bit in the DSI Host Interrupt and Status Register 1 (DSI\_ISR1) is asserted and an internal soft reset is generated to the DSI Host.

If the Timeout High-Speed Transmission Interrupt Enable (TOHSTXIE) bit of the DSI Host Interrupt Enable Register 1 (DSI\_IER1) is set, an interrupt is generated.

#### Low-power reception contention detection

The timeout is configured in the Low-Power Reception Timeout Counter (LPRX\_TOCNT) field of the DSI Host Timeout Counter Configuration Register 1 (DSI\_TCCR1). A 16-bit counter measures the time during which the Low-Power reception is active.

If that counter reaches the value defined by the Low-Power Reception Timeout Counter (LPRX\_TOCNT) field of the DSI Host Timeout Counter Configuration Register 1 (DSI\_TCCR0), the Timeout Low-Power Reception (TOLPRX) bit in the DSI Host Interrupt and Status Register 1 (DSI\_ISR1) is asserted and an internal soft reset is generated to the DSI Host.

If the Timeout Low-Power Reception Interrupt Enable (TOLPRXIE) bit of the DSI Host Interrupt Enable Register 1 (DSI\_IER1) is set, an interrupt is generated. Once the software gets notified by the interrupt, it must reset the D-PHY by de-asserting and asserting the Digital Enable (DEN) bit of the DSI Host PHY Control Register (DSI\_PCTLR).

## 29.8.2 Peripheral response timeout counters

A peripheral may not immediately respond correctly to some received packets. For example, a peripheral receives a read request, but due to its architecture cannot access the RAM for a while. It may be because the panel is being refreshed and takes some time to respond. In this case, set a timeout to ensure that the host waits long enough so that the device is able to process the previous data before receiving the new data or responding correctly to new requests.

[Table 178](#) lists the events belonging to various categories having an associated timeout for peripheral response.

**Table 178. List of events of different categories of the PRESP\_TO counter**

Category	Event
Items implying a BTA PRESP_TO	Bus Turn-Around
READ requests indicating a PRESP_TO (replicated for HS and LP)	(0x04) Generic read, no parameters short (0x14) Generic read, 1 parameter short (0x24) Generic read, 2 parameters short (0x06) DCS read, no parameters short
WRITE requests indicating a PRESP_TO (replicated for HS and LP)	(0x03) Generic short write, no parameters short (0x13) Generic short write, 1 parameter short (0x23) Generic short write, 2 parameters short (0x29) Generic long write long (0x05) DCS short write, no parameters short (0x15) DCS short write, 1 parameter short (0x39) DCS long write/write_LUT, Command packet long (0x37) Set maximum return packet size

The DSI Host ensures that, on sending an event that triggers a timeout, the D-PHY switches to the Stop state and a counter starts running until it reaches the value of that timeout. The link remains in the LP-11 state and unused until the timeout ends, even if there are other events ready to be transmitted.

Figures [187](#) to [189](#) illustrate the flow of counting in the PRESP\_TO counter for the three categories listed in [Table 178](#).

Figure 187. Timing of PRESP\_TO after a bus turn-around

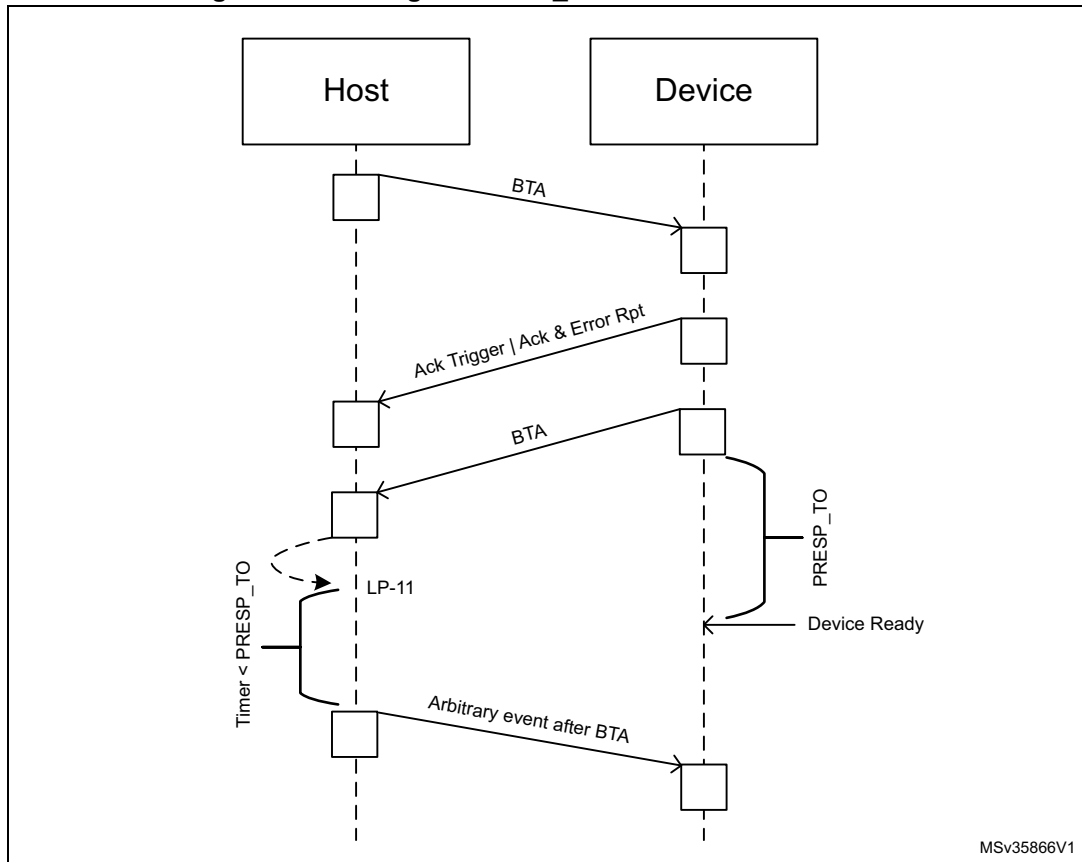


Figure 188. Timing of PRESP\_TO after a Read Request (HS or LP)

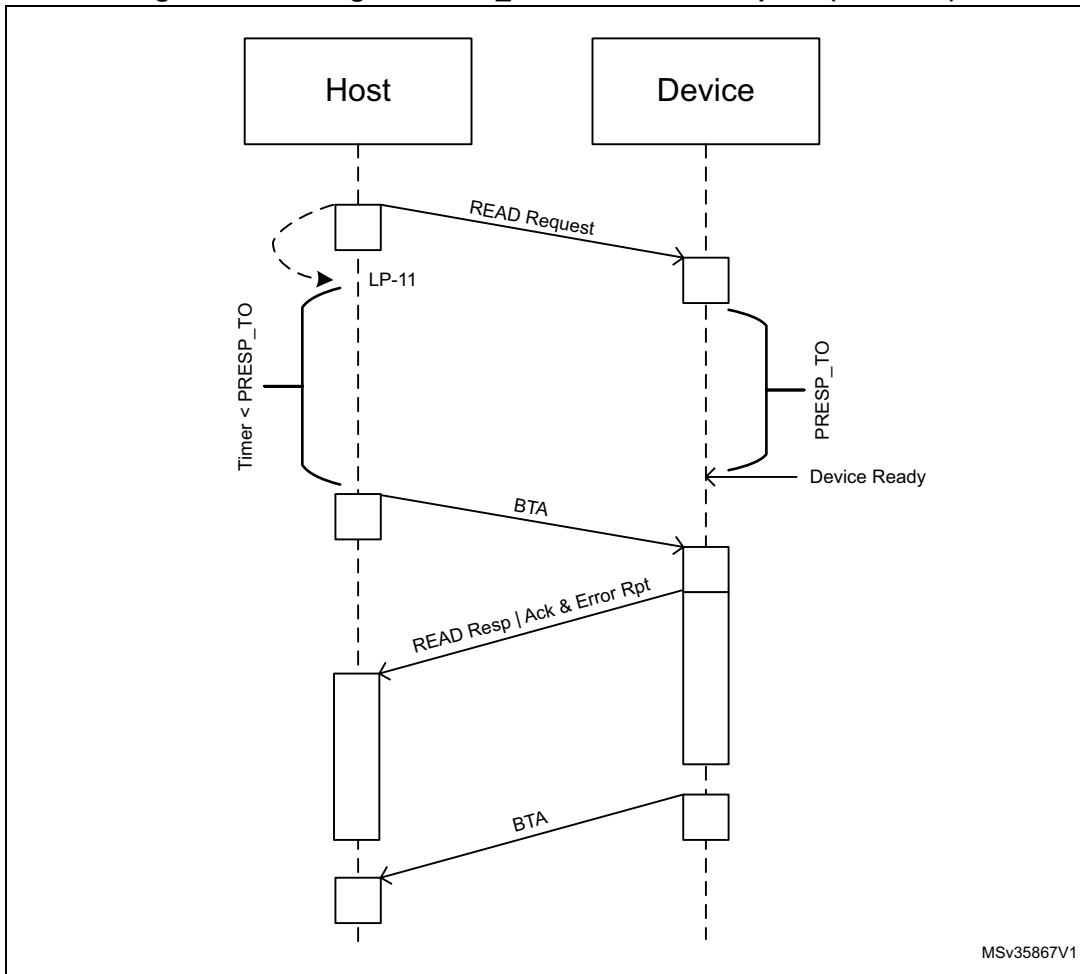


Figure 189. Timing of PRESP\_TO after a Write Request (HS or LP)

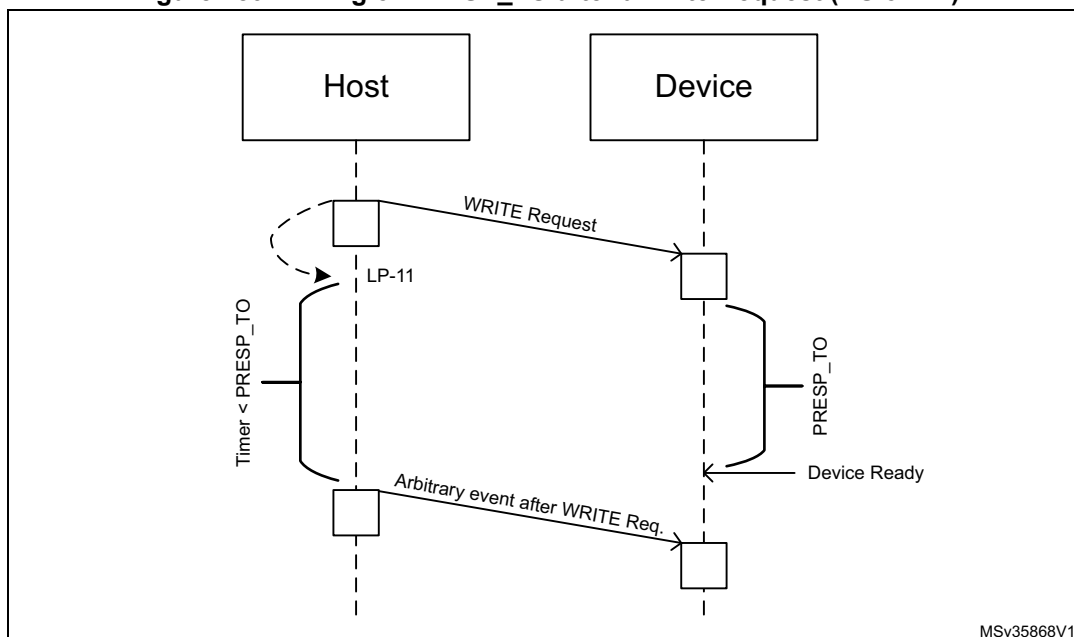


Table 179 describes the fields used for the configuration of the PRESP\_TO counter.

Table 179. PRESP\_TO counter configuration

Description	Register	Field	
Period for which the DSI Host keeps the link still	After sending a High-Speed read operation	DSI_TCCR1	HSRD_TOCNT
	After sending a Low-Power read operation	DSI_TCCR2	LPRD_TOCNT
	After completing a Bus-Turn-Around (BTA)	DSI_TCCR5	BTA_TOCNT
Period for which the DSI Host keeps the link inactive	After sending a High-Speed write operation	DSI_TCCR3	HSWR_TOCNT
	After sending a Low-Power write operation	DSI_TCCR4	LPWR_TOCNT

The values in these registers are measured in number of cycles of the Lane byte clock. These registers are only used in Command mode because in Video mode, there is a rigid timing schedule to be met to keep the display properly refreshed and it must not be broken by these or any other timeouts. Setting a given timeout to 0 disables going into LP-11 state and timeout for events of that category.

The read and the write requests in High-Speed mode are distinct from the read and the write requests in Low-Power mode. For example, if HSRD\_TOCNT is set to zero and LPRD\_TOCNT is set to a non-zero value, a generic read with no parameters does not activate the PRESP\_TO counter in High-Speed, but it activates the PRESP\_TO in Low-Power.

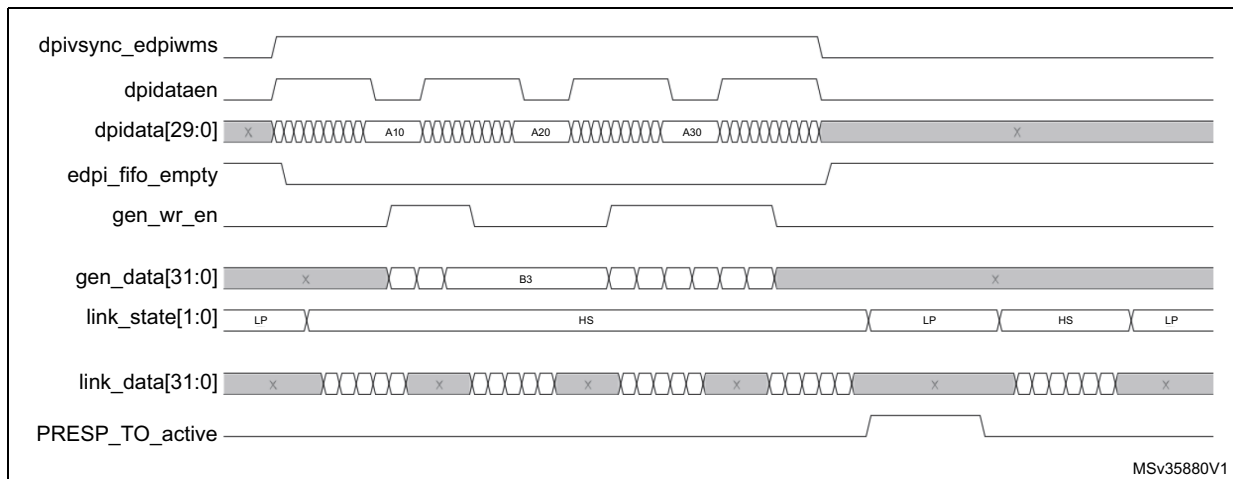
The DSI Host Timeout Counter Configuration Register 4 (DSI\_TCCR3) includes a special Presp mode (PM) bit to change the normal behavior of PRESP\_TO in Adaptive Command

mode for High-Speed write operation timeout. When set to 1, this bit allows the PRESP\_TO from HSWR\_TOCNT to be used only once, when both of the following conditions are met:

- the LTDC VSYNC signal rises and falls;
- the packets originated from the LTDC interface in Adapted Command mode are transmitted and its FIFO is empty again.

In this scenario, non-Adapted Command mode requests are not sent to the D-PHY, even if there is traffic from the Generic interface ready to be sent, returning them to the Stop state. When it happens, the PRESP\_TO counter is activated and only when it is completed, the DSI Host sends any other traffic that is ready, as illustrated in [Figure 190](#).

**Figure 190. Effect of Prep mode at 1**



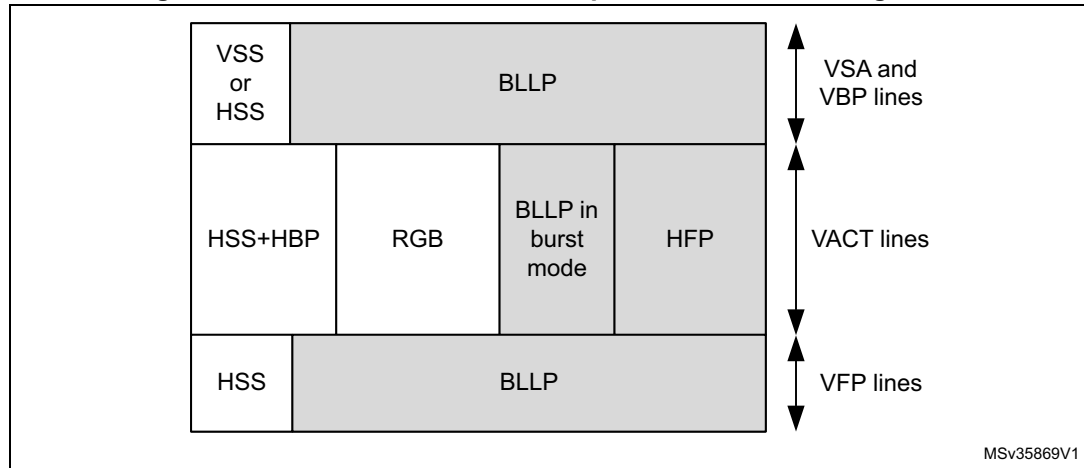


## 29.9 Functional description: transmission of commands

### 29.9.1 Transmission of commands in Video mode

The DSI Host supports the transmission of commands, both in High-Speed and Low-Power, while in Video mode. The DSI Host uses Blanking or Low-Power (BLLP) periods to transmit commands inserted through the APB Generic interface. Those periods correspond to the gray areas of [Figure 191](#).

**Figure 191. Command transmission periods within the image area**

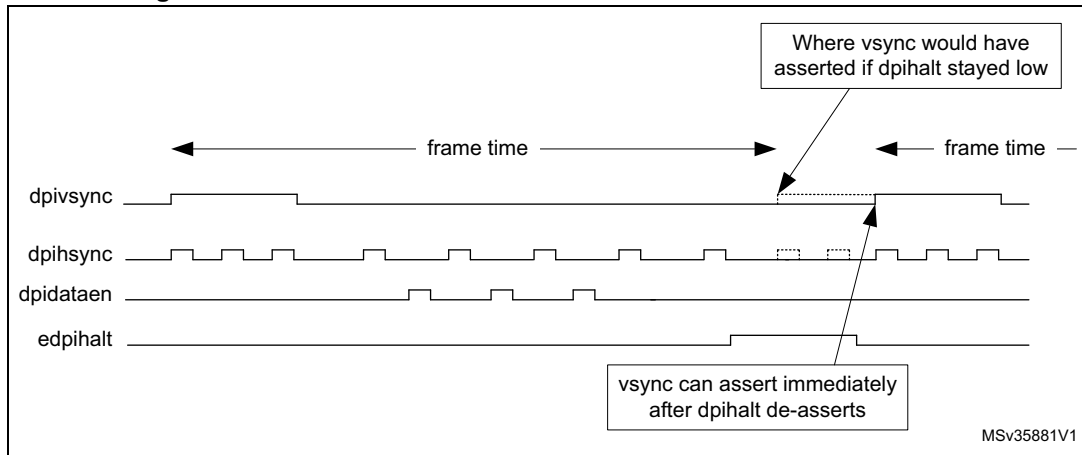


Commands are transmitted in the blanking periods after the following packets/states:

- Vertical Sync Start (VSS) packets, if the Video Sync pulses are not enabled
- Horizontal Sync End (HSE) packets, in the VSA, VBP, and VFP regions
- Horizontal Sync Start (HSS) packets, if the Video Sync pulses are not enabled in the VSA, VBP, and VFP regions
- Horizontal Active (HACT) state

Besides the areas corresponding to BLLP, large commands can also be sent during the last line of a frame. In that case, the line time for the Video mode is violated and the `edpiphalt` signal is set to request the DPI video timing signals to remain inactive. Only if a command does not fit into any BLLP area, it is postponed to the last line, causing the violation of the line time for the Video mode, as illustrated in [Figure 192](#).

**Figure 192. Transmission of commands on the last line of a frame**



Only one command is transmitted per line, even in the case of the last line of a frame but one command is possible for each line.

There can be only one command sent in Low-Power per line. However, one Low-Power command is possible for each line. In High-Speed, the DSI Host can send more than one command, as many as it determines to fit in the available time.

The DSI Host avoids sending commands in the last line because it is possible that the last line is shorter than the other ones. For instance, the line time ( $t_L$ ) could be half a cycle longer than the  $t_L$  on the LTDC interface, that is, each line in the frame taking half a cycle from time for the last line. This results in the last line being  $(\frac{1}{2} \text{ cycle}) \times (\text{number of lines} - 1)$  shorter than  $t_L$ .

The COLM and SHTDN bits of the DSI Wrapper Control Register (DSI\_WCR) are also able to trigger the sending of command packets. The commands are:

- Color mode ON
- Color mode OFF
- Shut Down Peripheral
- Turn On Peripheral

These commands are not sent in the VACT region. If the Low-Power Command Enable (LPCE) bit of the DSI Host Video mode Configuration Register (DSI\_VMCR) is set, these commands are sent in Low-Power mode.

In Low-Power mode, the Largest Packet Size (LPSIZE) field of the DSI Host Low-power mode Configuration Register (DSI\_LPMCR) is used to determine if these commands can be transmitted. It is assumed that Largest Packet Size (LPSIZE) is greater than or equal to four bytes (number of bytes in a short packet), because the DSI Host does not transmit these commands on the last line.

If the Frame Bus-Turn-Around Acknowledge Enable (FBTAAE) bit is set in the DSI Host Low-power mode Configuration Register (DSI\_LPMCR), a BTA is generated by DSI Host after the last line of a frame. This may coincide with a write command or a read command. In either case, the LTDC interface is halted until an acknowledge is received (control of the DSI bus is returned to the host).

### 29.9.2 Transmission of commands in Low-power mode

DSI Host can be configured to send the Low-Power commands during the High-Speed Video mode transmission.

To enable this feature, set the Low Power Command Enable (LPCE) bit of the DSI Host Video mode Configuration Register (DSI\_VMCR) to 1. In this case, it is necessary to calculate the time available, in bytes, to transmit a command in Low-Power mode to Horizontal Front Porch (HFP), Vertical Sync Active (VSA), Vertical Back Porch (VBP), and Vertical Front Porch (VFP) regions.

Bits 8 to 13 of the Video mode configuration register (DSI\_VMCR) register indicates if DSI Host can go to LP when in idle. If the Low-Power Command Enable (LPCE) bit is set and non-video packets are in queue, DSI Host ignores the Low-Power configuration and transmits Low-Power commands, even if it is not allowed to enter Low-Power mode in a specific region. After the Low-Power commands transmission, DSI Host remains in Low-Power until a sync event occurs.

For example, consider that the VFP is selected as High-Speed region (LPVFPE = 1'b0) with LPCE set as a command to transmit in Low-Power in the VFP region. This command is transmitted in Low-Power, and the line stays in Low-Power mode until a new HSS arrives.

#### Calculating the time to transmit commands in LP mode in the VSA, VBP, and VFP regions

The Largest Packet Size (LPSIZE) field of the DSI Host Low-Power mode Configuration Register (DSI\_LPMCR) indicates the time available (in bytes) to transmit a command in Low-Power mode (based on the escape clock) on a line during the VSA, VBP, and the VFP regions.

Calculation of Largest Packet Size (LPSIZE) depends on the used Video mode.

Figure 193 illustrates the timing intervals for the Video mode in Non-Burst with sync pulses, while Figure 194 refers to Video mode in Burst and Non-Burst with sync events.

Figure 193. LPSIZE for Non-Burst with sync pulses

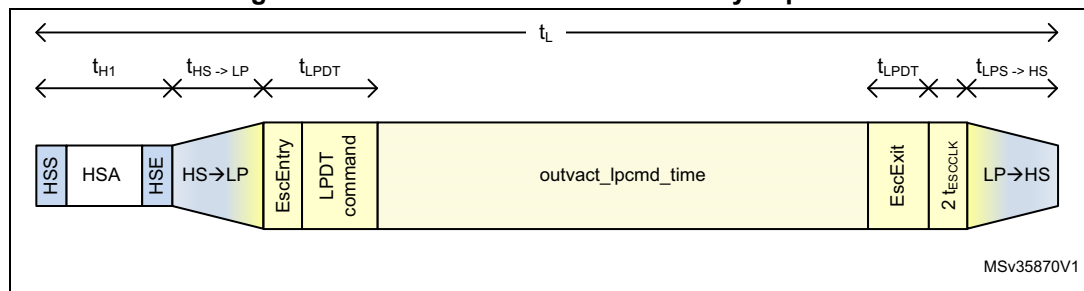
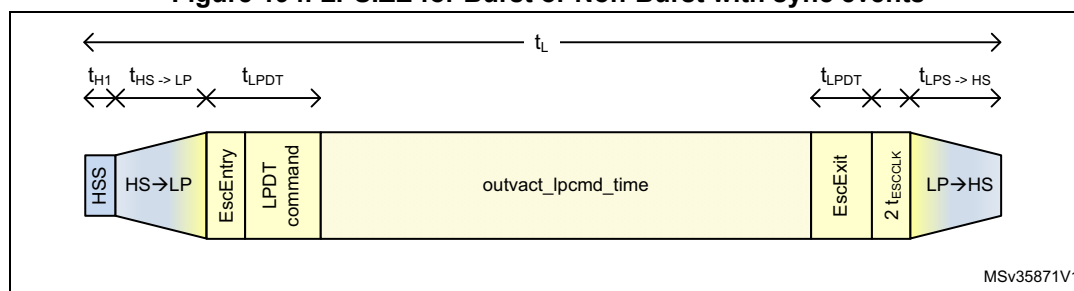


Figure 194. LPSIZE for Burst or Non-Burst with sync events



This time is calculated as follows:

$$\text{LPSIZE} = (t_L - (t_{H1} + t_{HS \rightarrow LP} + t_{LPDT} + t_{LPDT} + 2 t_{\text{ESCCLK}})) / (2 \times 8 \times t_{\text{ESCCLK}}), \text{ where}$$

- $t_L$  = line time;
- $t_{H1}$  = time of the HSA pulse for sync pulses mode (Figure 193) or time to send the HSS packet, including EoTp (Figure 194);
- $t_{HS \rightarrow LP}$  = time to enter the Low-Power mode;
- $t_{LP \rightarrow HS}$  = time to leave the Low-Power mode;
- $t_{LPDT}$  = D-PHY timing related with Escape mode Entry, LPDT Command, and Escape Exit. According to the D-PHY specification, this value is always 11 bits in LP (or 22 TX escape clock cycles);
- $t_{\text{ESCCLK}}$  = escape clock period as programmed in the TXECKDIV field of the DSI\_CCR register;
- $t_{\text{ESCCLK}}$  = delay imposed by the DSI Host implementation.

In the above equation, division by eight is done to convert the available time to bytes. Division by two is done because one bit is transmitted every two escape clock cycles. The Largest Packet Size (LPSIZE) field can be compared directly with the size of the command to be transmitted to determine if there is enough time to transmit the command. The maximum size of a command that can be transmitted in Low-Power mode is limited to 255 bytes by this field. You must program this register to a value greater than or equal to 4 bytes for the transmission of the DCTRL commands, such as shutdown and color in Low-Power mode.

Consider an example of a frame with 12.4  $\mu\text{s}$  per line and assume an escape clock frequency of 20 MHz and a lane bit rate of 800 Mbits. In this case, it is possible to send 124 bits in escape mode (that is, 124 bit = 12.4  $\mu\text{s}$  \* 20 MHz / 2). Still, you need to take into consideration the D-PHY protocol and PHY timings.

The following assumptions are made:

- lane byte clock period is 10 ns (800 Mbits per Lane);
- escape clock period is 50 ns (DSI\_CCR.TXECKDIV = 5);
- video is transmitted in Non-Burst mode with sync pulses bounded by HSS and HSE packets;
- DSI is configured for two lanes;
- D-PHY takes 180 ns to transit from Low-Power to High-Speed mode (DSI\_DLTCR.LS2HS\_TIME = 18);
- D-PHY takes 200 ns to transit from High-Speed to Low-Power mode (DSI\_DLTCR.HS2LP\_TIME = 20);
- $t_{HSA} = 420$  ns.

In this example, a 13-byte command can be transmitted as follows:

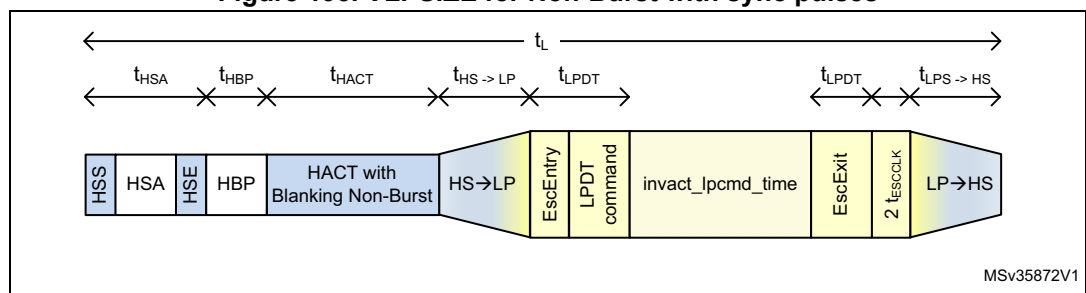
$$LPSIZE = (12.4 \mu s - (420 ns + 180 ns + 200 ns + (22 \times 50 ns + 2 \times 50 ns))) / (2 \times 8 \times 50 ns) = 13 \text{ bytes.}$$

### Calculating the Time to Transmit Commands in Low-Power mode in HFP region

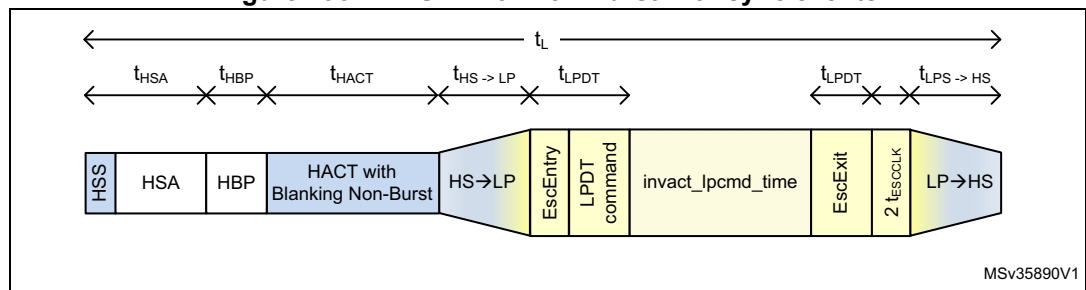
The VACT Largest Packet Size (VLPSIZE) field of the DSIHOST Low-Power mode Configuration Register (DSI\_LPMCR) indicates the time available (in bytes) to transmit a command in Low-Power mode (based on the escape clock) in the Vertical Active (VACT) region.

To calculate the value of VACT Largest Packet Size (VLPSIZE), consider the Video mode being used. [Figure 195](#) shows the timing intervals for Video mode in Non-Burst with sync pulses, [Figure 196](#) those for Video mode in Non-Burst with sync events, and [Figure 197](#) refers to the Burst Video mode.

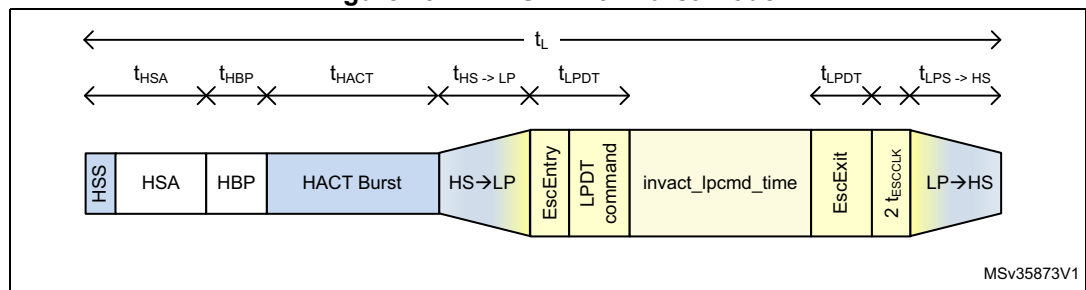
**Figure 195. VLPSIZE for Non-Burst with sync pulses**



**Figure 196. VLPSIZE for Non-Burst with sync events**



**Figure 197. VLPSIZE for Burst mode**



This time is calculated as follows:

$$VLPSIZE = (t_L - (t_{HSA} + t_{HBP} + t_{HACT} + t_{HS->LP} + t_{LP->HS} + t_{LPDT} + 2 t_{ESCCLK})) / (2 \times 8 \times t_{ESCCLK})$$

where

- $t_L$  = line time;
- $t_{HSA}$  = time of the HSA pulse (DSI\_VHSACR.HSA);
- $t_{HBP}$  = time of Horizontal back porch (DSI\_VHBPCR.HBP);
- $t_{HACT}$  = time of Video active. For Burst mode, the Video active is time compressed and is calculated as  $t_{HACT} = VPSIZE * Bytes\_per\_Pixel / Number\_Lanes * t_{Lane\_byte\_clk}$ ;
- $t_{ESCCLK}$  = escape clock period as programmed in TXECKDIV field of the DSI\_CCR register.

The VLPSIZE field can be compared directly with the size of the command to be transmitted to determine if there is time to transmit the command.

Consider an example of a frame with 16.4  $\mu$ s per line and assume an escape clock frequency of 20 MHz and a Lane bit rate of 800 Mbits/s. In this case, it is possible to send 420 bits in escape mode (that is, 164 bits = 16.4  $\mu$ s \* 20 MHz / 2). Still, since it is the Vertical Active region of the frame, take into consideration the HSA, HBP, and HACT timings apart from the D-PHY protocol and PHY timings. The following assumptions are made:

- number of active lanes is 4;
- Lane byte clock period (lanebyteclkperiod) is 10 ns (800 Mbits per Lane);
- escape clock period is 50 ns (DSI\_CCR.TXECKDIV = 5);
- D-PHY takes 180 ns to pass from Low-Power to High-Speed mode (DSI\_DLTCR.LP2HS\_TIME = 18);
- D-PHY takes 200 ns to pass from High-Speed to Low-Power mode (DSI\_DLTCR.HS2LP\_TIME = 20);
- $t_{HSA} = 420$  ns;
- $t_{HBP} = 800$  ns;
- $t_{HACT} = 12800$  ns to send 1280 pixel at 24 bpp;
- video is transmitted in Non-Burst mode;
- DSI Host is configured for four lanes.

In this example, consider that you send video in Non-Burst mode. The VLPSIZE is calculated as follows:

$$VLPSIZE = (16.4 \mu s - (420 \text{ ns} + 800 \text{ ns} + 12.8 \mu s + 180 \text{ ns} + 200 \text{ ns} + (22 \times 50 \text{ ns} + 2 \times 50 \text{ ns})) / (2 \times 8 \times 50 \text{ ns})) = 1 \text{ byte}$$

Only one byte can be transmitted in this period. A short packet (for example, generic short write) requires a minimum of four bytes. Therefore, in this example, commands are not sent in the VACT region.

If Burst mode is enabled, more time is available to transmit the commands in the VACT region, because HACT is time compressed.

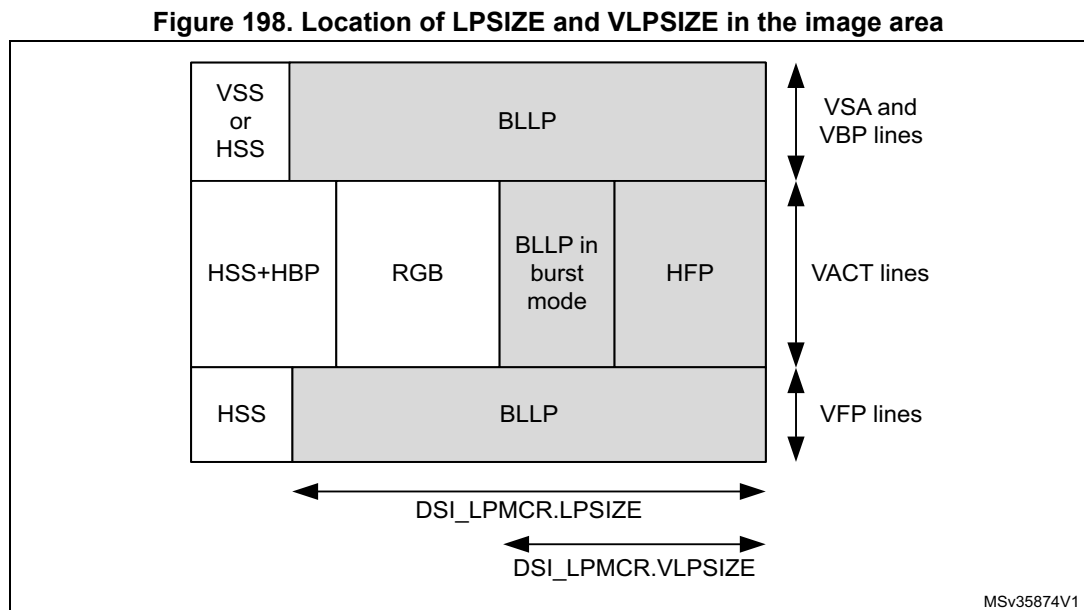
$$VLPSIZE = (16.4 \mu s - (420 \text{ ns} + 800 \text{ ns} + (1280 \times 3 / 4 \times 10 \text{ ns}) + 180 \text{ ns} + 200 \text{ ns} + (22 \times 50 \text{ ns} + 2 \times 50 \text{ ns})) / (2 \times 8 \times 50 \text{ ns})) = 5 \text{ bytes}$$

For Burst mode, the VLPSIZE is 5 bytes and then a 4-byte short packet can be sent.

### Transmission of commands in different periods

The LPSIZE and VLPSIZE fields allow a simple comparison to determine if a command can be transmitted in any of the BLLP periods.

Figure 198 illustrates the meaning of VLPSIZE and LPSIZE, matching them with the shaded areas and the VACT region.



### 29.9.3 Transmission of commands in High-speed

If the LPCE bit of the DSI\_VMCR register is 0, the commands are sent in High-Speed in Video mode. In this case, the DSI Host automatically determines the area where each command can be sent and no programming or calculation is required.

### 29.9.4 Read command transmission

The MRD\_TIME field of the DSI\_DLTCR register configures the maximum amount of time required to perform a read command in lane byte clock cycles, it is calculated as:

MRD\_TIME = Time to transmit the read command in Low-Power mode + Time to enter and leave Low-Power mode + Time to return the read data packet from the peripheral device.

The time to return the read data packet from the peripheral depends on the number of bytes read and the escape clock frequency of the peripheral, not the escape clock of the host. The MRD\_TIME field is used in both High-Speed and Low-Power mode to determine if there is time to complete a read command in a BLLP period.

In High-Speed mode (LPCE = 0), MRD\_TIME is calculated as follows:

$$MRD\_TIME = (t_{HS \rightarrow LP} + t_{LP \rightarrow HS} + t_{read} + 2 \times t_{BTA}) / \text{lanebyteclkperiod}$$

In Low-Power mode (LPCE = 1), MRD\_TIME is calculated as follows:

$MRD\_TIME = (t_{HS \rightarrow LP} + t_{LP \rightarrow HS} + t_{LPDT} + t_{iprd} + t_{read} + 2 \times t_{BTA}) / \text{lanebyteclkperiod}$ , where:

- $t_{HS \rightarrow LP}$  = Time to enter the Low-Power mode;
- $t_{LP \rightarrow HS}$  = Time to leave the Low-Power mode;
- $t_{LPDT}$  = D-PHY timing related to Escape mode entry, LPDT command, and Escape mode exit (according to the D-PHY specification, this value is always 11 bits in LP, or 22 TX escape clock cycles);
- $t_{iprd}$  = Read command time in Low-Power mode ( $64 \times \text{TX esc clock}$ );
- $t_{read}$  = Time to return the read data packet from the peripheral;
- $t_{BTA}$  = time to perform a bus turnaround (D-PHY dependent).

It is recommended to keep the maximum number of bytes read from the peripheral to a minimum to have sufficient time available to issue the read commands in a line time. Ensure that  $MRD\_TIME \times \text{Lane byte clock period}$  is less than  $LPSIZE \times 16 \times \text{escape clock period}$  of the host, otherwise, the read commands are dispatched on the last line of a frame. If it is necessary to read a large number of parameters ( $> 16$ ), increase the  $MRD\_TIME$  while the read command is being executed. When the read has completed, decrease the  $MRD\_TIME$  to a lower value.

If a read command is issued on the last line of a frame, the LTDC interface is halted and stays halted until the read command is in progress. The video transmission should be stopped during this period.

### 29.9.5 Clock lane in Low-power mode

To reduce the power consumption of the D-PHY, the DSI Host, when not transmitting in the High-Speed mode, allows the clock lane to enter into the Low-power mode. The controller automatically handles the transition of the clock lane from HS (Clock lane active sending clock) to LP state without direct intervention by the software. This feature can be enabled by configuring the DPCC and the ACR bits of the DSI\_CLCR register.

In the Command mode, the DSI Host can place the clock lane in the Low-Power mode when it does not have any HS packets to transmit.

In the Video mode (LTDC interface), the DSI Host controller uses its internal video and PHY timing configurations to determine if there is time available for the clock line to enter the Low-Power mode and not compromise the video data transmission of pixel data and sync events.

Along with a correct configuration of the Video mode (see [Section 29.5: Functional description: Video mode on LTDC interface](#)), the DSI Host needs to know the time required by the clock lane to go from High-Speed to Low-Power mode and viceversa. The values required can be obtained from the D-PHY specification: program the DSI\_CLTCR register with the following values:

- $HS2LP\_TIME$  = Time from HS to LP in clock lane / Byte clock period in HS (lanebyteclk)
- $LP2HS\_TIME$  = Time from LP to HS in clock lane / Byte clock period in HS (lanebyteclk)

Based on the programmed values, the DSI Host calculates if there is enough time for the clock lane to enter the Low-Power mode during inactive regions of the video frame.



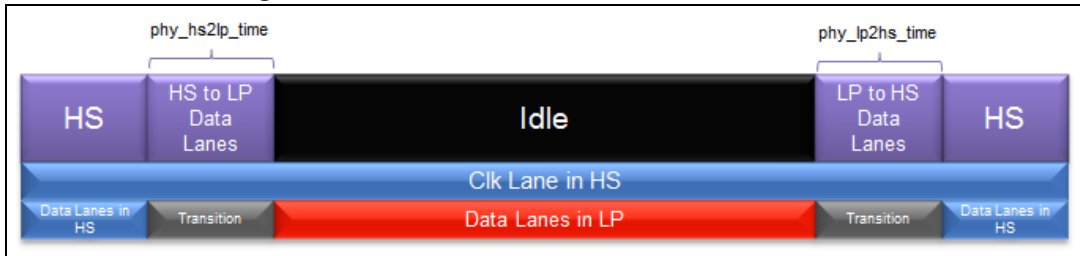
The DSI Host decides the best approach to follow regarding power saving out of the three possible scenarios:

- there is not enough time to go to the Low-Power mode. Therefore, blanking period is added as shown in [Figure 199](#);
- there is enough time for the data lanes to go to the Low-Power mode but not enough time for the clock lane to enter the Low-Power mode, see [Figure 200](#).
- there is enough time for both data lanes and clock lane to go to the Low-Power mode, as in [Figure 201](#).

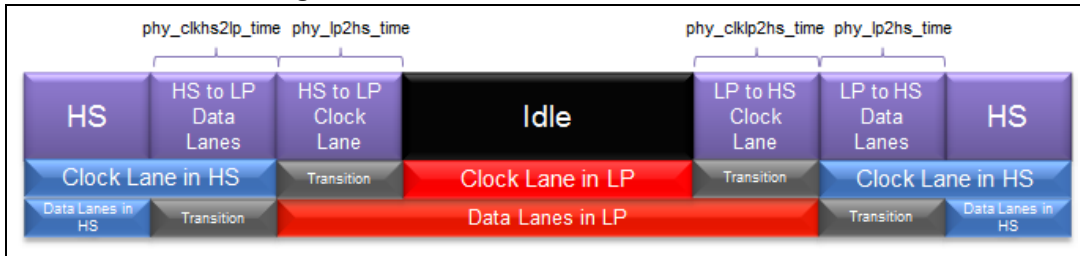
**Figure 199. Clock lane and data lanes in HS**



**Figure 200. Clock lane in HS and data lanes in LP**



**Figure 201. Clock lane and data lanes in LP**

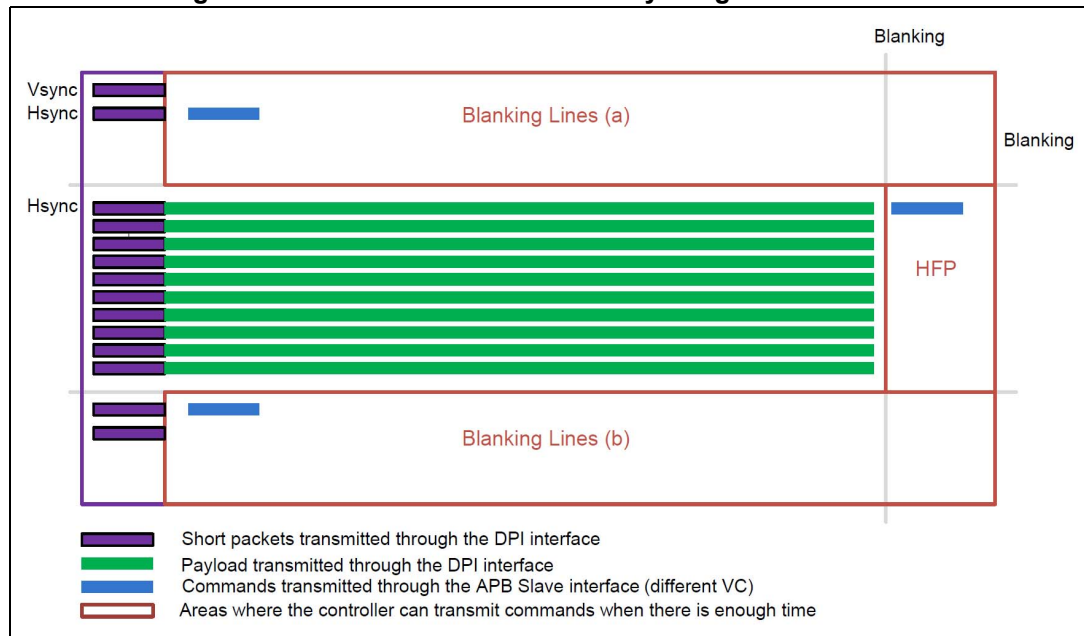


### 29.10 Functional description: virtual channels

The DSI Host supports choosing the Virtual Channel (VC) for use for each interface. Using multiple Virtual Channels, the system can address multiples displays at the same time, when each display has a different Virtual Channel identifier.

When the LTDC interface is configured for a particular Virtual Channel, it is possible to use the APB slave generic interface to issue the commands while the video stream is being transmitted. With this, it is possible to send the commands through the ongoing video stream, addressing different virtual channels and thus enable the interface with multiple displays. During the Video mode, the video stream transmission has the maximum priority. Therefore, the transmission of sideband packets such as the ones from the Generic interface are only transported when there is time available within the video stream transmission. The DSI Host identifies the available time periods and uses them to transport the Generic interface packets. *Figure 202* illustrates where the DSI Host inserts the packets from the APB Generic interface within the video stream transmitted by the LTDC interface.

**Figure 202. Command transmission by the generic interface**



It is also possible to address the multiple displays with only the Generic interface using different Virtual Channels. Because the Generic interface is not restricted to any particular Virtual Channel through configuration, it is possible to issue the packets with different Virtual Channels. This enables the interface to time multiplex the packets to be provided to the displays with different Virtual Channels.

You can use the following configuration registers to select the Virtual Channel ID associated with transmissions over the LTDC and APB slave generic interfaces:

- DSI\_LVCID.VCID field configures the Virtual Channel ID that is indexed to the Video mode packets using the LTDC interface.
- DSI\_GHCR register configures the Packet Header (which includes the Virtual Channel ID to be used) for transmissions using APB slave generic interface.
- DSI\_GVIDR.VCID field configures the Virtual Channel ID of the read responses to store and return to the Generic interface.

## 29.11 Functional description: video mode pattern generator

The Video mode pattern generator allows the transmission of horizontal/vertical color bar and D-PHY BER testing pattern without any stimuli.

The frame requirements must be defined in video registers that are listed in [Table 180](#).

**Table 180. Frame requirement configuration registers**

Register Name	Description
DSI Host Video mode Configuration Register	Video mode configuration
DSI Host Video Packet Configuration Register	Video packet size
DSI Host Video Chunks Configuration Register	Number of chunks
DSI Host Video Null Packet Configuration Register	Null packet size
DSI Host Video HSA Configuration Register	Horizontal sync active time
DSI Host Video HBP Configuration Register	Horizontal back porch time
DSI Host Video Line Configuration Register	Line time
DSI Host Video VSA Configuration Register	Vertical sync active period
DSI Host Video VBP Configuration Register	Vertical back porch period
DSI Host Video VFP Configuration Register	Vertical front porch period
DSI Host Video VA Configuration Register	Vertical resolution

### 29.11.1 Color bar pattern

The color bar pattern comprises eight bars for white, yellow, cyan, green, magenta, red, blue, and black colors.

Each color width is calculated by dividing the line pixel size (vertical pattern) or the number of lines (horizontal pattern) by eight. In the vertical color bar mode ([Figure 203](#)), each single color bar has a width of the number of pixels in a line divided by eight. In case the number of pixels in a line is not divisible by eight, the last color (black) contains the remaining.

In the horizontal color bar mode ([Figure 204](#)), each color line has a color width of the number of lines in a frame divided by eight. In case the number of lines in a frame is not divisible by eight, the last color (black) contains the remaining lines.

Figure 203. Vertical color bar mode

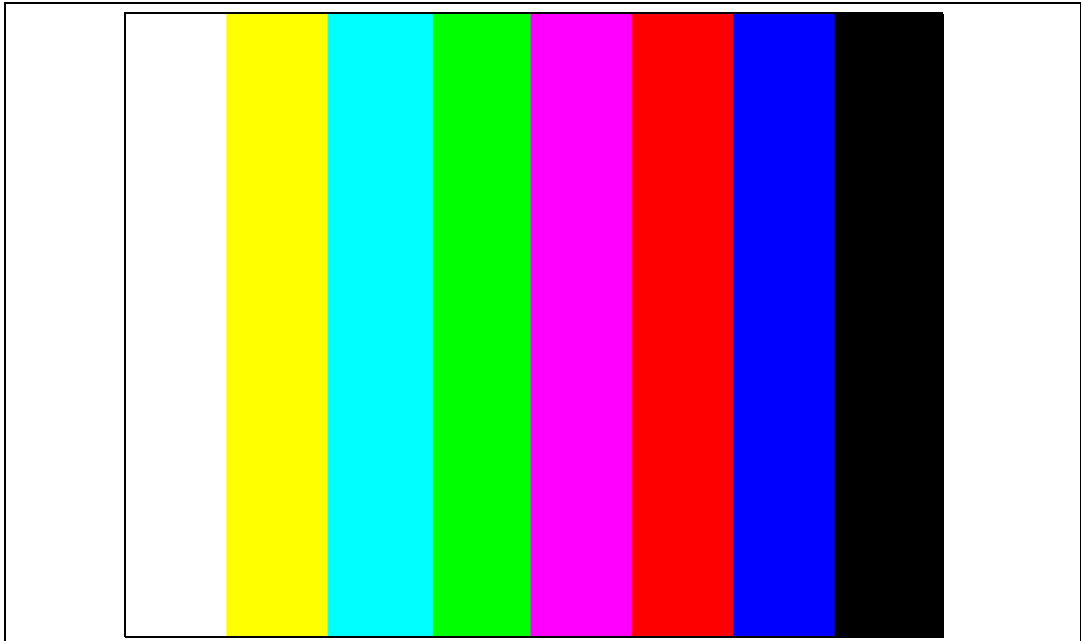
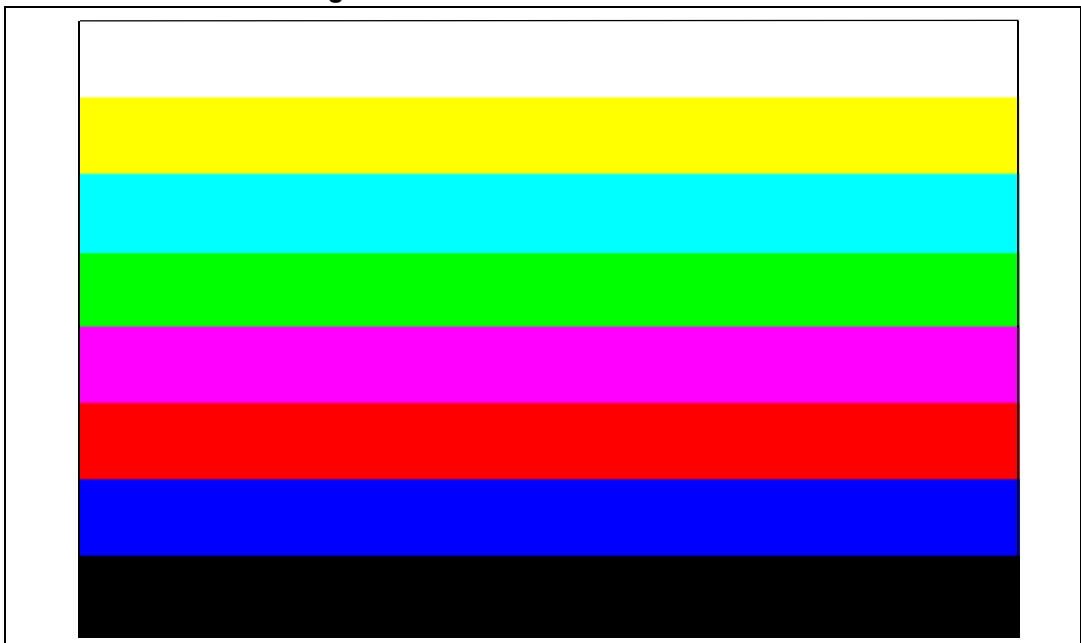


Figure 204. Horizontal color bar mode



### 29.11.2 Color coding

*Table 181* shows the RGB components used.

**Table 181. RGB components**

	White	Yellow	Cyan	Green	Magenta	Red	Blue	Black
R	High	High	Low	Low	High	High	Low	Low
G	High	High	High	High	Low	Low	Low	Low
B	High	Low	High	Low	High	Low	High	Low

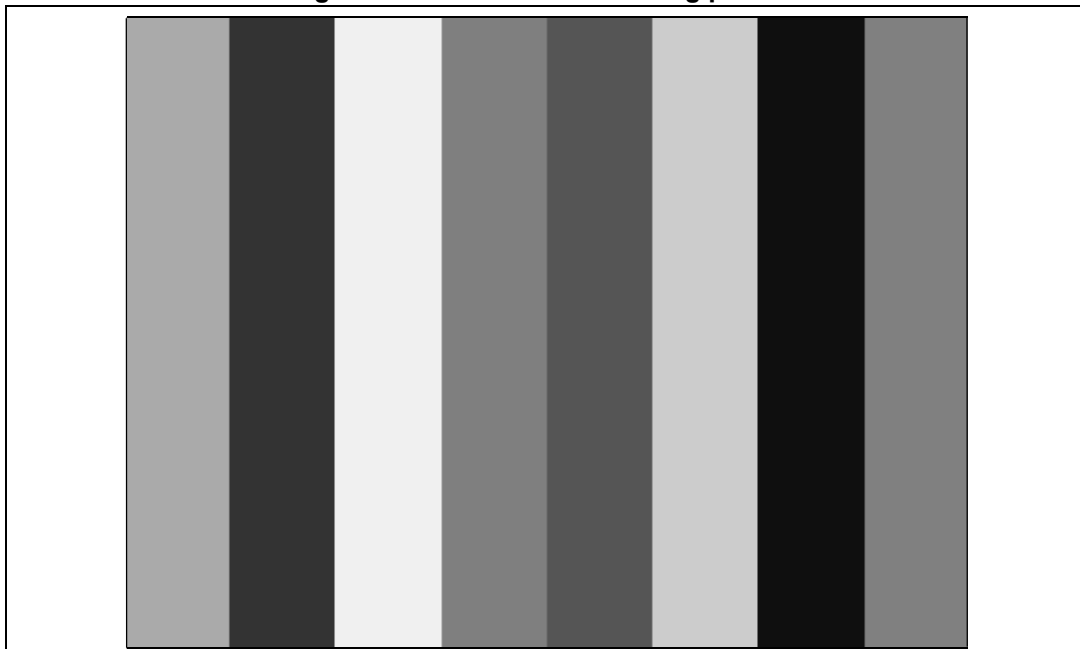
### 29.11.3 BER testing pattern

The BER testing pattern simplifies conformance testing. This pattern tests the RX D-PHY capability to receive the data correctly. The following data patterns are required:

- X bytes of 0xAA (high-frequency pattern, inverted);
- X bytes of 0x33 (mid-frequency pattern);
- X bytes of 0xF0 (low-frequency pattern, inverted);
- X bytes of 0x7F (lone 0 pattern);
- X bytes of 0x55 (high-frequency pattern);
- X bytes of 0xCC (mid-frequency pattern, inverted);
- X bytes of 0x0F (low-frequency pattern);
- Y bytes of 0x80 (lone 1 pattern).

In most cases, Y is equal to X. However, depending on line length and the color coding used, Y may be different from X. With RGB888 color coding and horizontal resolution in multiples of eight, the pattern shown in [Figure 205](#) appears on the DSI display.

**Figure 205. RGB888 BER testing pattern**



### 29.11.4 Video mode pattern generator resolution

Depending on the orientation, BER mode, and color coding, the smallest resolutions accepted by the Video mode pattern generator are:

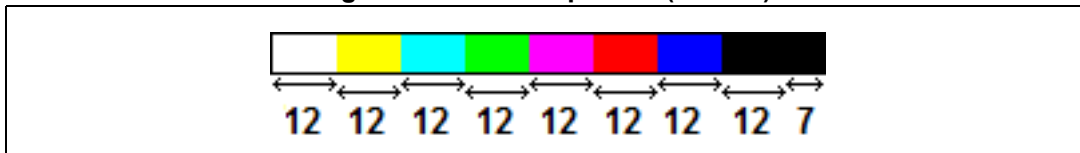
- BER mode: 8x8;
- horizontal color bar mode: 8x8;
- vertical color bar mode: 8x8.

#### Vertical pattern

The width of each color bar is determined by the division of horizontal resolution (pixels) for eight test pattern colors. If the horizontal resolution is not divisible by eight, the last color (black) is extended to fill the resolution.

In the example in [Figure 206](#), the horizontal resolution is 103.

Figure 206. Vertical pattern (103x15)



#### Horizontal pattern

The width of each color bar is determined by the division of the number of vertical resolution (lines) for eight test pattern colors. If the vertical resolution is not divisible by eight, the last color (black) will be extended to fill the resolution, as shown in [Figure 207](#).

Figure 207. Horizontal pattern (103x15)



## 29.12 Functional description: D-PHY management

The embedded MIPI® D-PHY is control directly by the DSI Host and is configured through the DSI Wrapper.

A dedicated PLL and a dedicated 1.2 V regulator are also embedded to supply the clock and the power supply to the DSI Host and D-PHY.

### 29.12.1 D-PHY configuration

The D-PHY configuration is carried out through the DSI Wrapper thanks to the DSI\_WPCR<sub>x</sub> registers.

#### Timing definition

The MIPI® D-PHY manages all the communication timing with dedicated timers. As all the timings are specified in nanoseconds (ns), it's mandatory to configure the Unit Interval Field to ensure the good duration of all the timings.

Unit Interval is configure through the DSI\_WPCR0.UIX4 field. This value defines the bit period in High-Speed mode in unit of 0.25ns. If this period is not a multiple of 0.25 ns, the value driven should be rounded down.

As an example, for a 300 Mbit/s link, the unit interval is 3.33 ns, so UIX4 shall be 13.33. In this case a value of 13 (0x0D) should be written.

#### Slew-rate and delay tuning on pins

To fine tune DSI communication, slew-rates and delays be fine tuned:

- slew-rate in High-Speed transmission on data lane and clock lane
- slew-rate in Low-Power transmission on data lane and clock lane
- transmission delay in High-Speed transmission on data land and clock lane

**Table 182. Slew-rate and delay tuning**

Function	Lane(s)	Value field in DSI_WPCR1
Slew-rate in High-Speed transmission	Clock lane	HSTXSRCCCL
	Data lanes	HSTXSRCDL
Slew-rate in Low-Power transmission	Clock lanes	LPSRCCL
	Data lanes	LPSRCDL
High-speed transmission delay	Clock lane	HSTXDCL
	Data lanes	HSTXDDL

The default values for all this parameters is 2'h00. All this values can be programmed only when the DSI is stopped (DSI\_WCR.DSIEN = 0 and CR.EN = 0).

#### Low-power reception filter tuning

The cut-off frequency of the low-pass on Low-Power receiver can be fine tuned through the LPRXFT field of the DSI\_WPCR1 register. The default values is 2'h00 and it can be programmed only when the DSI is stopped (CR.DSIEN = 0 and CR.EN = 0).

### Special Sdd Control

An additional current path can be activated on both clock lane and data lane to meet the Sdd<sub>TX</sub> parameter defined in the MIPI<sup>®</sup> D-PHY Specification.

This activation is done setting the SDDC bit of the DSI\_WPCR1 register.

### Custom lane configuration

To ease DSI integration, lane pins can be swapped and/or High-Speed signal can be inverted on a lane as described in [Table 183](#).

**Table 183. Custom lane configuration**

Function	Lane	Enable bit in DSI_WPCR0
Swap lane pins	Clock lane	SWCL
	Data lane 0	SWDL0
	Data lane 1	SWDL1
Invert High-Speed signal on lane	Clock lane	HSICL
	Data lane 0	HSIDL0
	Data lane 1	HSIDL1

### Custom timing configuration

Some of the MIPI<sup>®</sup> D-PHY timing can be tuned for specific purpose as described in [Table 184](#).

**Table 184. Custom timing parameters**

MIPI <sup>®</sup> timing	Enable bit in DSI_WPCR0	Configuration register	Field	Default value	Default duration
t <sub>CLK-POST</sub>	TCLKPOSTEN	DSI_WPCR4	TCLKPOST	200	100 ns + 120*UI
t <sub>L PX (Clock lane)</sub>	TLPXCEN	DSI_WPCR3	TLPXC	100	50 ns
t <sub>HS_EXIT</sub>	THSEXITEN		THSEXIT	200	100 ns + 40*UI
t <sub>L PX (Data lane)</sub>	TLPXDEN		TLPXD	100	50 ns
t <sub>HS-ZERO</sub>	THSZEROEN		THSZERO	175	175 ns + 8*UI
t <sub>HS-TRAIL</sub>	THSTRAIL	DSI_WPCR2	THSTRAIL	140	70 ns + 8*UI
t <sub>HS-PREPARE</sub>	THSPREPEN		THSPREP	126	63 ns + 12*UI
t <sub>CLK-ZERO</sub>	TCLKZEROEN		TCLKZERO	195	390 ns
t <sub>CLK-PREPARE</sub>	TCLKPREPEN		TCLKPREP	120	60 ns + 20*UI

All this values can be programmed only when the DSI is stopped (CR.DSIEN = 0 and CR.EN = 0).



### 29.12.2 Special D-PHY operations

The DSI Wrapper have some control bit to force the D-PHY in some particular state and/or behavior.

#### Forcing lane state

It's possible to force the data lane and/or the clock lane in TX Stop mode through the bits FTXSMDL and FTXSMCL of the DSI\_WPCR1 register.

Setting this bits causes the respective lane module to immediately jump in transmit control mode and to begin transmitting a stop state (LP-11).

This feature can be used to go back in TX ode after a wrong BTA sequence.

#### Forcing Low-Power receiver in Low-Power mode

The FLPRXLPM bit of the DSI\_WPCR1 register enables the Low-Power mode of the low power receiver (LPRX). When set, the LPRX operates in Low-Power mode all the time. When not set, the LPRX operates in Low-Power mode during ULPS only.

#### Disabling turn of data lane

When set, the TDDL bit of the DSI\_WPCR0 register forces the data lane to remain in reception mode even if a Bus Turn Around request (BTA) is received from the other side.

### 29.12.3 Special Low-power D-PHY functions

The embedded D-PHY offers two specific features to optimize consumption.

#### Pull-down on lanes

The D-PHY embedded pull-down on each lane to prevent from floating states when the lanes are unused.

When set, the PDEN bit of the DSI\_WPCR0 register enables the pull-down on the lanes.

#### Disabling contention detection on data lanes

The contention detector on the data lane can be turned off to lower the overall D-PHY consumption.

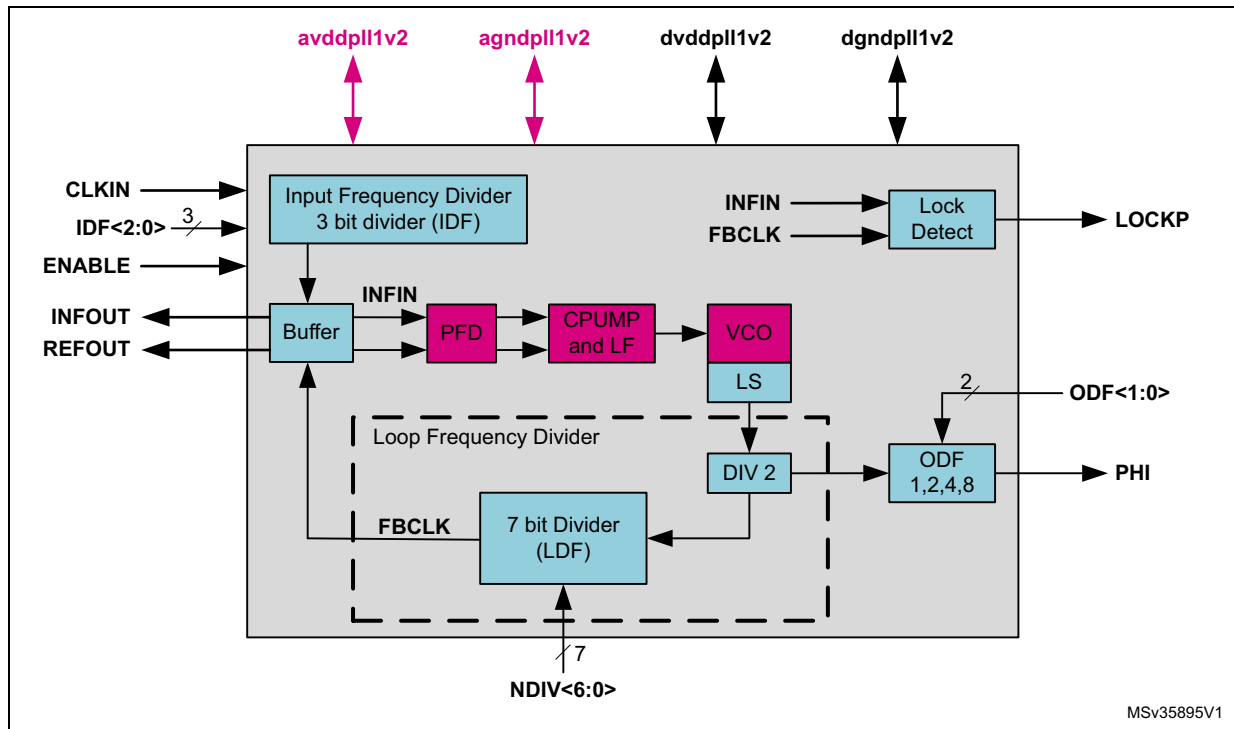
When set, the CDOFFDL bit of the DSI\_WPCR0 register disables the contention detection on data lanes.

This can be used in forward Escape mode to reduce the static power consumption.

### 29.12.4 DSI PLL control

The dedicated DSI PLL is controlled through the DSI Wrapper, as shown in *Figure 208* (analog blocks and signals in pink, digital signals in black, digital blocks in light blue).

Figure 208. PLL block diagram



The PLL output frequency is configured through the DSI\_WRPCR register fields. The VCO frequency and the PLL output frequency are calculated as follows:

$$F_{VCO} = (CLK_{IN} / IDF) * 2 * NDIV,$$

$$PHI = F_{VCO} / (2 * ODF)$$

where:

- CLK<sub>IN</sub> is in the range of 4 to 100 MHz;
- DSI\_WRPCR.NDIV is in the range of 10 to 125;
- DSI\_WRPCR.IDF is in the range of 1 to 7;
- INFIN is in the range of 4 to 25 MHz;
- F<sub>VCO</sub> is in the range of 500 MHz to 1 GHz;
- DSI\_WRPCR.ODF can be 1, 2, 4 or 8;
- PHI is in the range of 31.25 to 500 MHz.

The PLL is enabled setting the PLEN bit in the DSI\_WRPCR register.

Once the PLL is locked, the PLLIF bit is set in the DSI\_WISR. If the PLLIE bit is set in the DSI\_WIER, an interrupt is generated.

The PLL status (lock or unlock) can be monitored with the PLLLS flag in the DSI\_WISR register.

If the PLL gets unlocked, the PLLUIF bit of the DSI\_WISR is set. If the PLLUIE bit of the DSI\_WIER register is set, an interrupt is generated.

The DSI PLL setting can be changed only when the PLL is disabled.

### 29.12.5 Regulator control

The DSI regulator providing the 1.2 V is controlled through the DSI Wrapper.

The regulator is enabled setting the REGEN bit of the DSI\_WRPCR register.

Once the regulator is ready, the RRIF bit of the DSI\_WISR register is set. If the RRIE bit of the DSI\_WIER register is set, an interrupt is generated.

The regulator status (ready or not) can be monitored with the RRS flag in the DSI\_WISR register.

Note that the D-PHY has no separated Power ON control bit. The power ON/OFF of the D-PHY is done directly enabling the 1.2 V regulator.

When the 1.2 V regulator is disabled, the 3.3 V part of the D-PHY is automatically powered OFF.

## 29.13 Functional description: interrupts and errors

The interrupts can be generated either by the DSI Host or by the DSI Wrapper.  
All the interrupts are merged in one interrupt lane going to the Interrupt Controller.

### 29.13.1 DSI wrapper interrupts

An interrupt can be produced on the following events:

- tearing effect event;
- end of refresh;
- PLL locked;
- PLL unlocked;
- regulator ready.

Separate interrupt enable bits are available for flexibility.

**Table 185. DSI wrapper interrupt requests**

Interrupt event	Event flag in DSI_WISR	Enable control bit in DSI_WIER
Tearing effect	TEIF	TEIE
End of refresh	ERIF	ERIE
PLL locked	PLLLIF	PLLLIE
PLL unlocked	PLLUIF	PLLUIE
Regulator ready	RRIF	RRIE

### 29.13.2 DSI host interrupts and errors

The DSI\_ISR0 and DSI\_ISR1 registers are associated with error condition reporting. These registers can trigger an interrupt to inform the system about the occurrence of errors.

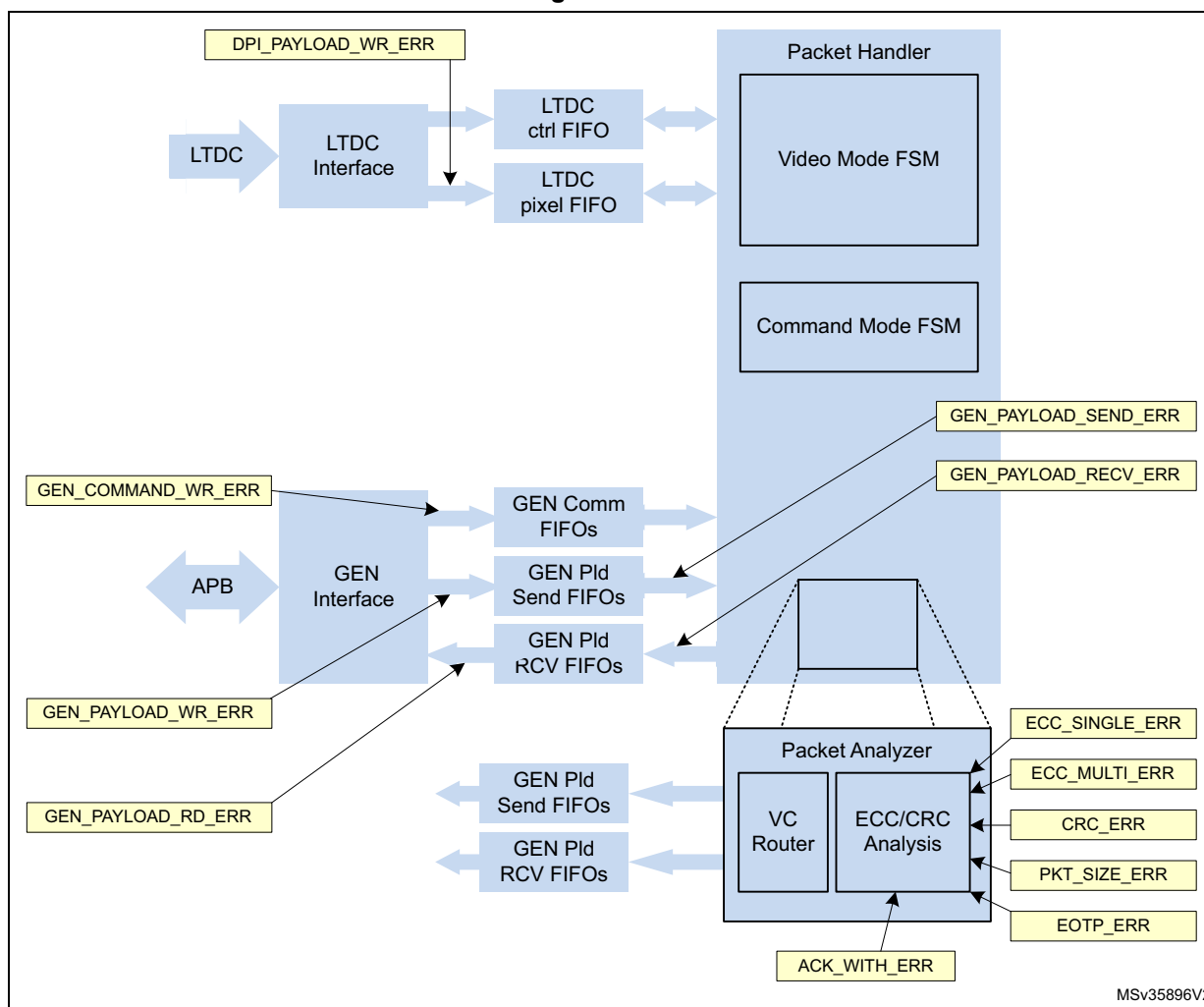
The DSI Host has one interrupt line that is set high when an error occurs in either the DSI\_ISR0 or the DSI\_ISR1 register.

The triggering of the interrupt can be masked by programming the mask registers DSI\_IER0 and DSI\_IER1. By default all errors are masked. When any bit of these registers is set to 1, it enables the interrupt for a specific error. The error bit is always set in the respective DSI\_ISR register. The DSI\_ISR0 and DSI\_ISR1 registers are always cleared after a read operation. The interrupt line is cleared if all registers that caused the interrupt are read.

The interrupt force registers (DSI\_FIR0 and DSI\_FIR1) are used for test purposes, and they allow triggering the interrupt events individually without the need to activate the conditions that trigger the interrupt sources; this is because it is extremely complex to generate the stimuli for that purpose. This feature also facilitates the development and testing of the software associated with the interrupt events. Setting any bit of these registers to 1 triggers the corresponding interrupt.

The light yellow boxes in *Figure 209* illustrate the location of some of the errors.

**Figure 209. Error sources**



*Table 186* explains the reasons that set off these interrupts and also explains how to recover from these interrupts.

**Table 186. Error causes and recovery**

DSI host interrupt & Status register	Bit	Name	Cause of the error	Recommended method of handling the error
0	20	PE4	The D-PHY reports the LP1 contention error. The D-PHY host detects the contention while trying to drive the line high.	Recover the D-PHY from contention. Reset the DSI Host and transmit the packets again. If this error is recurrent, carefully analyze the connectivity between the Host and the Device.
0	19	PE3	D-PHY reports the LP0 contention error. The D-PHY Host detects the contention while trying to drive the line low.	Recover the D-PHY from contention. Reset the DSI Host and transmit the packets again. If this error is recurrent, carefully analyze the connectivity between the Host and the Device.

**Table 186. Error causes and recovery (continued)**

DSI host interrupt & Status register	Bit	Name	Cause of the error	Recommended method of handling the error
0	18	PE2	The D-PHY reports the False Control Error. The D-PHY detects an incorrect line state sequence in lane 0 lines.	Device does not behave as expected, communication with the Device is not properly established. This is an unrecoverable error. Reset the DSI Host and the D-PHY. If this error is recurrent, analyze the behavior of the Device.
0	17	PE1	The D-PHY reports the LPDT Error. The D-PHY detects that the LDPT did not match a multiple of 8 bits.	The data reception is not reliable. The D-PHY recovers but the received data from the Device might not be reliable. It is recommended to reset the DSI Host and repeat the RX transmission.
0	16	PE0	The D-PHY reports the Escape Entry Error. The D-PHY does not recognize the received Escape Entry Code.	The D-PHY Host does not recognize the Escape Entry Code. The Transmission is ignored. The D-PHY Host recovers but the system should repeat the RX reception.
0	15	AE15	This error is directly retrieved from Acknowledge with Error packet. The Device detected a protocol violation in the reception.	Refer to the display documentation. When this error is active, the Device should have another read-back command that reports additional information about this error. Read the additional information and take appropriate actions.
0	14	AE14	The Acknowledge with Error packet contains this error. The Device chooses to use this bit for error report.	Refer to the Device documentation regarding possible reasons for this error and take appropriate actions.
0	13	AE13	The Acknowledge with Error packet contains this error. The Device reports that the transmission length does not match the packet length.	Possible reason for this is multiple errors present in the packet header (more than 2), so the error detection fails and the Device does not discard the packet. In this case, the packet header is corrupt and can cause decoding mismatches. Transmit the packets again. If this error is recurrent, carefully analyze the connectivity between the Host and the Device.
0	12	AE12	The Acknowledge with Error packet contains this error. The Device does not recognize the VC ID in at least one of the received packets.	Possible reason for this is multiple errors present in the packet header (more than 2), so the error detection fails and the Device does not discard the packet. In this case, the packet header is corrupt and can cause decoding mismatches. Transmit the packets again. If this error is recurrent, carefully analyze the connectivity between the Host and the Device.

Table 186. Error causes and recovery (continued)

DSI host interrupt & Status register	Bit	Name	Cause of the error	Recommended method of handling the error
0	11	AE11	The Acknowledge with Error packet contains this error. The Device does not recognize the data type of at least one of the received packets.	Check the Device capabilities. It is possible that there are some packets not supported by the Device. Repeat the transmission.
0	10	AE10	The Acknowledge with Error packet contains this error. The Device detects the CRC errors in at least one of the received packets.	Some of the long packets, transmitted after the last Acknowledge request, might contain the CRC errors in the payload. If the payload content is critical, transmit the packets again. If this error is recurrent, carefully analyze the connectivity between the Host and the Device.
0	9	AE9	The Acknowledge with Error packet contains this error. The Device detects multi-bit ECC errors in at least one of the received packets.	The Device does not interpret the packets transmitted after the last Acknowledge request. If the packets are critical, transmit the packets again. If this error is recurrent, carefully analyze the connectivity between the Host and the Device.
0	8	AE8	The Acknowledge with Error packet contains this error. The Device detects and corrects the 1 bit ECC error in at least one of the received packets.	No action is required. The Device acknowledges the packet. If this error is recurrent, analyze the signal integrity or the noise conditions of the link.
0	7	AE7	The Acknowledge with Error packet contains this error. The Device detects the Line Contention through LP0/LP1 detection.	This error might corrupt the Low-Power data reception and transmission. Ignore the packets and transmit them again. The Device recovers automatically. If this error is recurrent, check the Device capabilities and the connectivity between the Host and Device. Refer to section 7.2.1 of the DSI Specification 1.1.
0	6	AE6	The Acknowledge with Error packet contains this error. The Device detects the False Control Error.	The device detects one of the following: – The LP-10 (LP request) is not followed by the remainder of a valid escape or turnaround sequence. – The LP-01 (HS request) is not followed by a bridge state (LP-00). The D-PHY communications are corrupted. This error is unrecoverable. Reset the DSI Host and the D-PHY. Refer to the section 7.1.6 of the DSI Specification 1.1.

Table 186. Error causes and recovery (continued)

DSI host interrupt & Status register	Bit	Name	Cause of the error	Recommended method of handling the error
0	5	AE5	The Acknowledge with Error packet contains this error. The display timeout counters for a HS reception and LP transmission expire.	It is possible that the Host and Device timeout counters are not correctly configured. The Device HS_TX timeout should be shorter than the Host HS_RX timeout. Host LP_RX timeout should be longer than the Device LP_TX timeout. Check and confirm that the Host configuration is consistent with the Device specifications. This error is automatically recovered, although there is no guarantee that all the packets in the transmission or reception are complete. For additional information about this error, see section 7.2.2 of the DSI Specification 1.1.
0	4	AE4	The Acknowledge with Error packet contains this error. The Device reports that the LPDT is not aligned in an 8-bit boundary	There is no guarantee that the Device properly receives the packets. Transmit the packets again. For additional information about this error, see section 7.1.5 of the DSI Specification.
0	3	AE3	The Acknowledge with Error packet contains this error. The Device does not recognize the Escape mode Entry command.	The Device does not recognize the Escape mode Entry code. Check the Device capability. For additional information about this error, see section 7.1.4 of the DSI Specification. Repeat the transmission to the Device.
0	2	AE2	The Acknowledge with Error packet contains this error. The Device detects the HS transmission did not end in an 8-bit boundary when the EoT sequence is detected.	There is no guarantee that the Device properly received the packets. Re-transmission should be performed. Transmit the packets again. For additional information about this error, see section 7.1.3 of the DSI Specification 1.1.
0	1	AE1	The Acknowledge with Error packet contains this error. The Device detects that the SoT leader sequence is corrupted.	The Device discards the incoming transmission. Re-transmission should be performed by the Host. For additional information about this error, see section 7.1.2 of the DSI Specification 1.1.
0	0	AE0	The Acknowledge with Error packet contains this error. The Device reports that the SoT sequence is received with errors but synchronization can still be achieved.	The Device is tolerant to single bit and some multi-bit errors in the SoT sequence but the packet correctness is compromised. If the packet content was important, transmit the packets again. For additional information about this error, see section 7.1.1 of the DSI Specification 1.1.



Table 186. Error causes and recovery (continued)

DSI host interrupt & Status register	Bit	Name	Cause of the error	Recommended method of handling the error
1	12	GPRXE	An overflow occurs in the Generic read FIFO.	The Read FIFO size is not correctly dimensioned for the maximum read-back packet size. Configure the Device to return the read data with a suitable size for the Host dimensioned FIFO. Data stored in the FIFO is corrupted. Reset the DSI Host and repeat the read procedure.
1	11	GPRDE	An underflow occurs in the Generic read FIFO.	System does not wait for the read procedure to end and starts retrieving the data from the FIFO. The read data is requested before it is fully received. Data is corrupted. Reset the DSI Host and repeat the read procedure. Check that the read procedure is completed before reading the data through the APB interface.
1	10	GPTXE	An underflow occurs in the Generic write payload FIFO.	The system writes the packet header before the respective packet payload is completely loaded into the payload FIFO. This error is unrecoverable, the transmitted packet is corrupted. Reset the DSI Host and repeat the write procedure.
1	9	GPWRE	An overflow occurs in the Generic write payload FIFO.	The payload FIFO size is not correctly dimensioned to store the total payload of a long packet. Data stored in the FIFO is corrupted. Reset the DSI Host and repeat the write procedure.
1	8	GCWRE	An overflow occurs in the Generic command FIFO.	The command FIFO size is not correctly dimensioned to store the total headers of a burst of packets. Data stored in the FIFO is corrupted. Reset the DSI Host and repeat the write procedure.
1	7	LPWRE	An overflow occurs in the DPI pixel payload FIFO.	The controller FIFO dimensions are not correctly set up for the operating resolution. Check the Video mode configuration registers. They should be consistent with the LTDC video resolution. The pixel data sequence is corrupted. Reset the DSI Host and re-initiate the Video transmission.
1	6	EOTPE	Host receives a transmission that does not end with an End of Transmission packet.	This error is not critical for the data integrity of the received packets. Check if the Device supports the transmission of EoTp packets.

Table 186. Error causes and recovery (continued)

DSI host interrupt & Status register	Bit	Name	Cause of the error	Recommended method of handling the error
1	5	PSE	Host receives a transmission that does not end in the expected by boundaries.	The integrity of the received data cannot be guaranteed. Reset the DSI Host and repeat the read procedure.
1	4	CRCE	Host reports that a received long packet has a CRC error in its payload.	The received payload data is corrupted. Reset the DSI Host and repeat the read procedure. If this error is recurrent, check the DSI connectivity link for the noise levels.
1	3	ECCME	Host reports that a received packet contains multiple ECC errors.	The received packet is corrupted. The DSI Host ignores all the following packets. The DSI Host should repeat the read procedure.
1	2	ECCSE	Host reports that a received packet contains a single bit error.	This error is not critical because the DSI Host can correct the error and properly decode the packet. If this error is recurrent, check the DSI connectivity link for signal integrity and noise levels.
1	1	TOLPRX	Host reports that the configured timeout counter for the Low-Power reception has expired.	Once the configured timeout counter ends, the DSI Host automatically resets the controller side and recovers to normal operation. Packet transmissions happening during this event are lost. If this error is recurrent, check the timer configuration for any issue. This timer should be greater than the maximum Low-Power transmission generated by the Device.
1	0	TOHSTX	Host reports that the configured timeout counter for the High-Speed transmission has expired.	Once the configured timeout counter ends, the DSI Host automatically resets the controller side and recovers to normal operation. Packet transmissions happening during this event are lost. If this error is recurrent, check the timer configuration for any issue. This timer should be greater than the maximum High-Speed transmission bursts generated by the Host.
DSI Wrapper	10	PLLUF	The PLL of the D-PHY has unlocked.	This error can be critical. The graphical subsystem shall be reconfigured and restarted.

## 29.14 Programming procedure

To operate DSI Host, you must be familiar with the MIPI® DSI specification. Every software programmable register is accessible through the APB interface.

### 29.14.1 Programming procedure overview

The programming procedure for Video mode or Adapted Command mode must respect the following order:

1. Configure the RCC (refer to the RCC chapter)
  - Enable clock for DSI and LTDC
  - Configure LTDC PLL, turn it ON and wait for its lock
2. Optionally configure the GPIO (if tearing effect requires GPIO usage for example)
3. Optionally valid the ISR
4. Configure the LTDC (refer to the LTDC chapter)
  - Program the panel timings
  - Enable the relevant layers
5. Turn on the DSI regulator and wait for the regulator ready as described in [Section 29.12.5](#)
6. Configure the DSI PLL, turn it ON and wait for its lock as described in [Section 29.12.4](#)
7. Configure the D-PHY parameters in the DSI Host and the DSI Wrapper to define D-PHY configuration & timing as detailed in [Section 29.14.2](#)
8. Configure the DSI Host timings as detailed in [Section 29.14.3](#)
9. Configure the DSI Host Flow Control and DBI interface as detailed in [Section 29.14.4](#)
10. Configure the DSI Host LTDC interface as detailed in [Section 29.14.5](#)
11. Configure the DSI Host for Video mode as detailed in [Section 29.14.6](#) or Adapted Command mode as detailed in [Section 29.14.7](#)
12. Enable the D-PHY setting the DEN bit of the DSI\_PCTLR
13. Enable the D-PHY clock lane setting the CKEN bit of the DSI\_PCTLR
14. Enable the DSI Host setting the EN bit of the DSI\_CR
15. Enable the DSI Wrapper setting the DSIEN bit of the DSI\_WCR
16. Optionally send DCS commands through the APB generic interface to configure the display
17. Enable the LTDC in the LTDC
18. Start the LTDC flow through the DSI Wrapper (CR.LTDCEN = 1)

In Video mode, the data streaming starts as soon as the LTDC is enabled.

In Adapted Command mode, the frame buffer update is launched as soon as the CR.LTDCEN bit is set.

### 29.14.2 Configuring the D-PHY parameters

The D-PHY requires a specific configuration prior starting any communications. The configuration parameters are stored either in the DSI Host or the DSI Wrapper.

### Configuring the D-PHY parameters in the DSI wrapper

The DSI wrapper can be used to fine tune either timing or physical parameters of the D-PHY. This operation is not required for a standard usage of the D-PHY. All the fields and parameters are described in the register description of the DSI wrapper.

Only one field is mandatory to properly start the D-PHY: the Unit Interval multiplied by 4 (UIX4) field of the DSI wrapper PHY Configuration Register 1 (DSI\_WPCR0).

This field defines the bit period in High-Speed mode in unit of 0.25 ns, and is used as a timebase for all the timings managed by the D-PHY.

If the link is working at 600Mbit/s, the unit interval shall be 1.667 ns, i.e 6.667 ns when multiplied by four. When rounded down, a value of 6 must be written in the UIX4 field of the DSI\_WPCR0 register.

### Configuring the D-PHY parameters in the DSI host

The DSI Host stores the configuration of D-PHY timing parameters and number of lanes.

The following fields must be configured prior to any startup:

- Number of data lanes in the DSI\_PCONFR register
- Automatic clock lane control (ACR) in the DSI\_CLCR register
- Clock control (DPCC) in the DSI\_CLCR register
- Time for LP/HS and HS/LP transitions for both clock lane and data lanes in DSI\_CLTCR and DSI\_DLTCR registers
- Stop wait time in the DSI\_PCONFR register

#### 29.14.3 Configuring the DSI host timing

All the protocol timing shall be configured in the DSI host.

##### Clock divider configuration

Two clocks are generated internally

- Timeout Clock;
- TX Escape Clock.

The timeout clock is used as the timing unit in the configuration of HS to LP and LP to HS transition error. Its division factor is configured by the Timeout Clock Division (TOCKDIV) field of the DSI Host Clock Control Register (DSI\_CCR).

The TX Escape clock is used in Low-Power transmission. Its division factor is configured by the TX Escape Clock Division (TXECKDIV) field of the DSI Host Clock Control Register (DSI\_CCR) relatively to the lanebyteclock. Its typical value shall be around 20MHz.

##### Timeout configuration

The timings for timeout management as described in [Section 29.8](#) are configured in the DSI Host Timeout Counter Configuration Registers (DSI\_TCCR0 to DSI\_TCCR5).

#### 29.14.4 Configuring flow control and DBI interface

The flow control is configured thanks to the DSI Host Protocol Configuration Register (DSI\_PCR). The configuration parameters are the following

- CRC Reception Enable (CRCRXE bit)
- ECC Reception Enable (ECCRXE bit)
- BTA Enable (BTAE bit)
- EoTp Reception Enable (ETRXE bit)
- EoTp Transmission Enable (ETTXE bit)

Their values depends on the protocol to be used for the communication with the DSI display.

The Virtual Channel ID used for the generic DBI interface shall be configured by the Virtual Channel ID (VCID) field of the DSI Host Generic VCID Register (DSI\_GVCIDR).

All the DCS command, depending on their type, can be transmitted or received either in High-Speed or Low-Power. For each of them, a dedicated configuration bit shall be programmed in the DSI Host Command mode Configuration Register (DSI\_CMCR).

Acknowledge request for packet or tearing effect event shall also be configured in the DSI Host Command mode Configuration Register (DSI\_CMCR).

#### 29.14.5 Configuring the DSI host LTDC interface

As the DSI Host is interface to the system through the LTDC for Video mode or Adapted Command mode, the DSI Wrapper perform a low level interfacing in between.

The parameter programmed into the DSI wrapper must be aligned with the parameters programmed into the LTDC and the DSI host.

The following fields must be configured:

- Virtual Channel ID in the Virtual Channel ID (VCID) field of the DSI Host LTDC VCID Register (DSI\_LVCIDR).
- Color Coding (COLC) field of the DSI Host LTDC Color Coding Register (DSI\_LCOLCR) and the Color Muxing (COLMUX) in the DSI Wrapper Configuration Register (DSI\_WCFGR).
- If loose packets are used for 18-bit mode, the Loosely Packet Enable (LPE) bit of the DSI Host LTDC Color Coding Register (DSI\_LCOLCR) must be set.
- The HSYNC polarity in the HSync Polarity (HSP) bit of the DSI Host LTDC Polarity Configuration Register (DSI\_LPCR).
- The VSYNC polarity in the VSync Polarity (VSP) bit of the DSI Host LTDC Polarity Configuration Register (DSI\_LPCR) and in the VSync Polarity (VSPOL) bit of the DSI Wrapper Configuration Register (DSI\_WCFGR).
- The DATA ENABLE polarity Data Enable Polarity (DEP) bit of the DSI Host LTDC Polarity Configuration Register (DSI\_LPCR).

### 29.14.6 Configuring the video mode

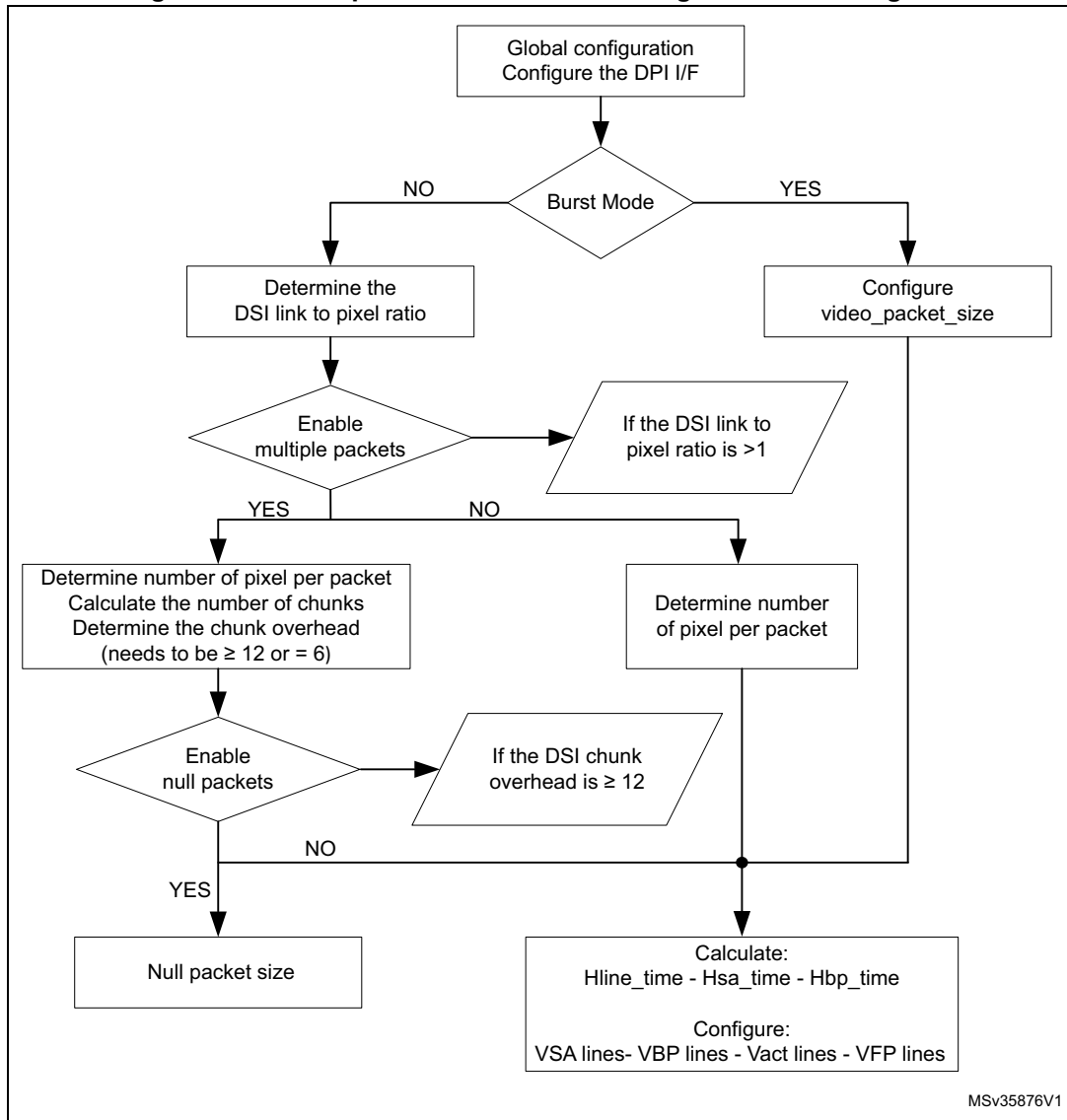
The video mode configuration shall defines the behavior of the controller in Low-power for command transmission, the type of video transmission (burst or non-burst mode) and the panel horizontal and vertical timing:

- Select the video transmission mode to define how the processor requires the video line to be transported through the DSI link.
  - Configure the Low-Power transitions in the DSI\_VMCR to define the video periods which are permitted to go to Low-Power if there is time available to do so.
  - Configure if the controller should request the peripheral acknowledge message at the end of frames (DSI\_VMCR.FBTAAE).
  - Configure if commands are to be transmitted in Low-Power (DSI\_VMCR.LPE).
- Select the video mode type
  - Burst mode:
    - Configure the video mode type (DSI\_VMCR.VMT) with value 2'b1x.
    - Configure the video packet size (DSI\_VPCR.VPSIZE) with the size of the active line period, measured in pixels.
    - The registers DSI\_VCCR and DSI\_VNPCR are ignored by the DSI Host.
  - Non-Burst mode:
    - Configure the video mode type (DSI\_VMCR.VMT) with 2'b00 to enable the transmission of sync pulses or with 2'b01 to enable the transmission of sync events.
    - Configure the video packet size (DSI\_VPCR.VPSIZE) with the number of pixels to be transmitted in a single packet. Selecting this value depends on the available memory of the attached peripheral, if the data is first stored, or on the memory you want to select for the FIFO in DSI Host.
    - Configure the number of chunks (DSI\_VCCR.NUMC) with the number of packets to be transmitted per video line. The value of VPSIZE \* NUMC is the number of pixels per line of video, except if NUMC is 0, which disables the multi-packets. If you set it to 1, there is still only one packet per line, but it can be part of a chunk, followed by a null packet.
    - Configure the null packet size (DSI\_VNPCR.NPSIZE) with the size of null packets to be inserted as part of the chunks. Setting it to 0 disables null packets.
- Define the video horizontal timing configuration as follows:
  - Configure the horizontal line time (DSI\_VLCR.HLINE) with the time taken by a LTDC video line measured in cycles of lane byte clock (for a clock lane at 500 MHz the lane byte clock period is 8 ns). When the periods of LTDC clock and lane byte clock are not multiples, the value to program the DSI\_VLCR.HLINE needs to be rounded. A timing mismatch is introduced between the lines due to the rounding of configuration values. If the DSI Host is configured not to go to Low-Power, this timing divergence accumulates on every line, introducing a significant amount of mismatch towards the end of the frame. The reason for this is that the DSI Host cannot re-synchronize on every new line because it transmits the blanking packets when the Horizontal Sync event occurs on the LTDC interface. However, the accumulated mismatch should become extinct on the last line of a frame, where, according to the DSI specification, the link should always return to Low-Power regaining synchronization, when a new frame starts on a vertical sync event. If the accumulated timing mismatch is greater than the time in Low-Power on the last

- line, a malfunction occurs. This phenomenon can be avoided by configuring the DSI Host to go to Low-Power once per line.
- Configure the horizontal sync duration (DSI\_VHSACR.HSA) with the time taken by a LTDC Horizontal Sync Active period measured in cycles of lane byte clock (normally a period of 8 ns).
  - Configure the horizontal back porch duration (DSI\_VHBPCR.HBP) with the time taken by the LTDC Horizontal Back Porch period measured in cycles of lane byte clock (normally a period of 8 ns). Special attention should be given to the calculation of this parameter.
  - Define the vertical line configuration:
    - Configure the vertical sync duration (DSI\_VVSACR.VSA) with the number of lines existing in the LTDC Vertical Sync Active period.
    - Configure the vertical back porch duration (DSI\_VVBPCR.VBP) with the number of lines existing in the LTDC Vertical Back Porch period.
    - Configure the vertical front porch duration (DSI\_VVFPCR.VFP) with the number of lines existing in the LTDC Vertical Front Porch period.
    - Configure the vertical active duration (DSI\_VVACR.VA) with the number of lines existing in the LTDC Vertical Active period.

*Figure 210* illustrates the steps for configuring the DPI packet transmission.

Figure 210. Video packet transmission configuration flow diagram



**Example of video configuration**

The following is an example of video packet transmission configuration:

Video resolution:

- PCLK period = 50 ns
- HSA = 8 PCLK
- HBP = 8 PCLK
- HACT = 480 PCLK
- HFP = 24 PCLK
- VSA = 2 lines
- VBP = 2 lines
- VACT = 640 lines
- VFP = 4 lines



Configuration steps:

- Video transmission mode configuration:
  - a) Configure the Low-Power transitions:  
DSI\_VMCR[13:8] = 6'b111111, to enable LP in all video period.
  - b) DSI\_VMCR.FBTAAE = 1, for the DSI Host to request an acknowledge response message from the peripheral at the end of each frame.
- To use the Burst mode, follow these steps:  
DSI\_VMCR.VMT = 2'b1x  
DSI\_VPCR.VPSIZE = 480
- Horizontal timing configuration:
  - $DSI\_VLCR.HLINE = (HSA + HBP + HACT + HFP) * (PCLK \text{ period} / \text{Clk Lane Byte Period}) = (8 + 8 + 480 + 24) * (50 / 8) = 3250$
  - $DSI\_VHSACR.HSA = HSA * (PCLK \text{ period} / \text{Clk Lane Byte Period}) = 8 * (50 / 8) = 50$
  - $DSI\_VHDPGR.HBP = HBP * (PCLK \text{ period} / \text{Clk Lane Byte Period}) = 8 * (50 / 8) = 50$
- Vertical line configuration:
  - DSI\_VVSACR.VSA = 2
  - DSI\_VVBPCR.VBP = 2
  - DSI\_VVFPCR.VFP = 4
  - DSI\_VVACR.VA = 640

### 29.14.7 Configuring the adapted command mode

The Adapted Command mode requires the following parameters to be configured:

- Command Size (CMD\_SIZE) field of the DSI Host LTDC Command Configuration Register (DSI\_LCCR) to define the maximum allowed size for a Write Memory Command.
- The tearing effect source (TESRC) and optionally tearing effect polarity (TEPOL) bits of the DSI Wrapper Configuration Register (DSI\_WCFGR).
- The Automatic Refresh (AR) bit of the DSI Wrapper Configuration Register (DSI\_WCFGR) if the display needs to be updated automatically each time a tearing effect event is received.

### 29.14.8 Configuring the video mode pattern generator

DSI Host can transmit a color bar pattern without horizontal/vertical color bar and D-PHY BER testing pattern without any kind of stimuli.

*Figure 211* shows the programming sequence to send a test pattern:

1. Configure the DSI\_MCR register to enable video mode. Configure the video mode type using DSI\_VMCR.VMT.
2. Configure the DSI\_LCOLCR register.
3. Configure the frame using registers shown in *Figure 212*, where the gray area indicated the transferred pixels).
4. Configure the pattern generation mode (DSI\_VMCR.PGM) and the pattern orientation (DSI\_VMCR.PGO), and enable it (DSI\_VMCR.PGE).

**Figure 211. Programming sequence to send a test pattern**

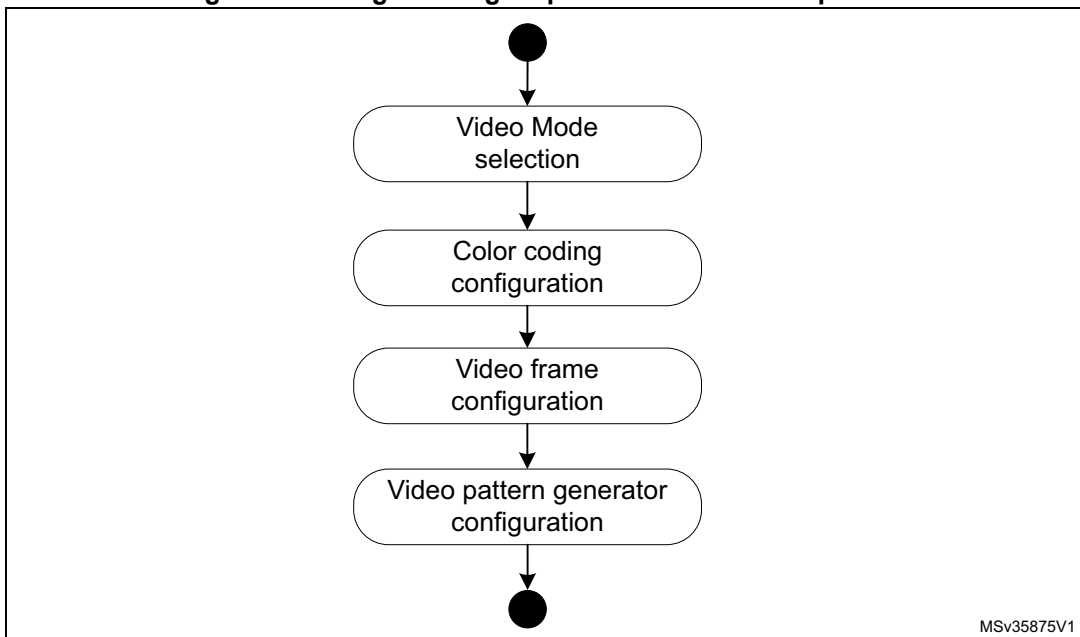
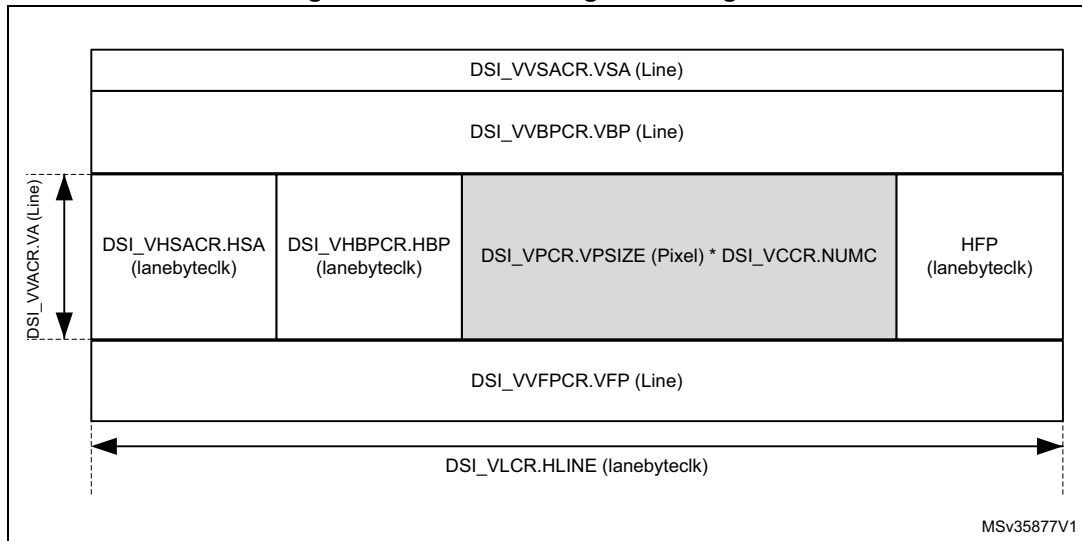


Figure 212. Frame configuration registers



Note: The number of pixels of payload is restricted to a multiple of a value provided in [Table 176](#).

### 29.14.9 Managing ULPM

There are two ways to configure the software to enter and exit the ULPM:

- Enter and Exit the ULPM with the D-PHY PLL running. This is a faster process.
- Enter and Exit the ULPM with the D-PHY PLL turned off. This is a more efficient process in terms of power consumption.

#### Clock management for ULPM sequence

The ULPM management state machine is working on the lanebyteclock provided by the D-PHY.

Because the D-PHY is providing the lanebyteclock only when the clock lane is not in ULPM state, it is mandatory to switch the lanebyteclock source of the DSI Host before starting the ULPM mode entry sequence.

The lanebyteclock source is controlled by the RCC. It can be

- the lanebyteclock provided by the D-PHY (for all modes except ULPM)
- a clock generated by the system PLL (for ULPM)

#### Process flow to enter the ULPM

Implement the process described in detail in the following procedure to enter the ULPM on both clock lane and data lanes:

1. Verify the initial status of the DSI Host:
  - DSI\_PCTLR[2:1] = 2'h3
  - DSI\_WRPCR.PLLEN = 1'h1 & DSI\_WRPCR.REGEN = 1'h1
  - DSI\_PUCR[3:0] = 4'h0
  - DSI\_PTTTCR[3:0] = 4'h0
  - Verify that all active lanes are in Stop state and the D-PHY PLL is locked:  
 One-lane configuration: DSI\_PSR[6:4] = 3'h3 & DSI\_PSR[1] = 1'h0 & DSI\_WISR.PLLS = 1'h1  
 Two-lanes configuration: DSI\_PSR[8:4] = 5'h1B & DSI\_PSR[1] = 1'h0 & DSI\_WISR.PLLS = 1'h1
2. Switch the lanebyteclock source in the RCC from D-PHY to system PLL
3. Set DSI\_PUCR[3:0] = 4'h5 to enter ULPM in the data and the clock lanes.
4. Wait until the D-PHY active lanes enter into ULPM:
  - One-lane configuration: DSI\_PSR[6:1] = 6'h00
  - Two-lanes configuration: DSI\_PSR[8:1] = 8'h00

The DSI Host is now in ULPM.
5. Turn off the D-PHY PLL by setting DSI\_WRPCR.PLLEN = 1'b0

### Process flow to exit the ULPM

Implement the process flow described in the following procedure to exit the ULPM on both clock lane and data lanes:

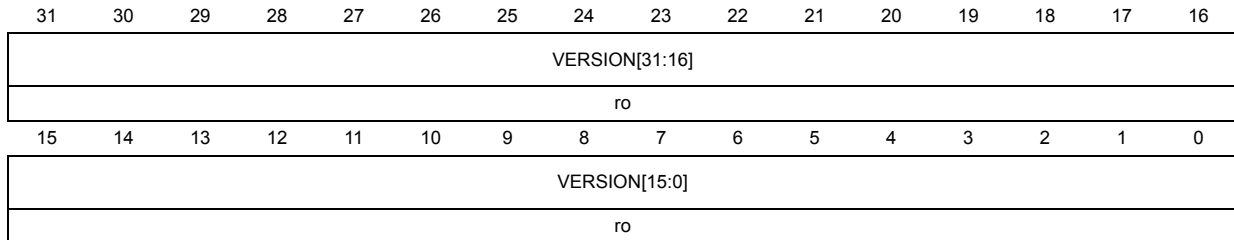
1. Verify that all active lanes are in ULPM:
  - One-lane configuration: DSI\_PSR[6:1] = 6'h00
  - Two-lanes configuration: DSI\_PSR[8:1] = 8'h00
2. Turn on the D-PHY PLL by setting DSI\_WRPCR.PLLEN = 1'b1.
3. Wait until D-PHY PLL locked
  - DSI\_WISR.PLLS = 1'b1
4. Without de-asserting the ULPM request bits, assert the Exit ULPM bits by setting DSI\_PUCR[3:0] = 4'hF.
5. Wait until all active lanes exit ULPM:
  - One-lane configuration:  
DSI\_PSR[5] = 1'b1  
DSI\_PSR[3] = 1'b1
  - Two-lanes configuration:  
DSI\_PSR[8] = 1'b1  
DSI\_PSR[5] = 1'b1  
DSI\_PSR[3] = 1'b1
6. Wait for 1 ms.
7. De-assert the ULPM requests and the ULPM exit bits by setting DSI\_PUCR [3:0] = 4'h0.
8. Switch the lanbyteclock source in the RCC from system PLL to D-PHY
9. The DSI Host is now in Stop state and the D-PHY PLL is locked:
  - One-lane configuration:  
DSI\_PSR[6:4] = 3'h3  
DSI\_PSR[1] = 1'h0  
DSI\_WRPCR.PLLEN = 1'b1
  - Two-lanes configuration:  
DSI\_PSR[8:4] = 5'h1B  
DSI\_PSR[1] = 1'h0  
DSI\_WRPCR.PLLEN = 1'b1

## 29.15 DSI Host registers

### 29.15.1 DSI Host Version Register (DSI\_VR)

Address offset: 0x0000

Reset value: 0x3133 302A

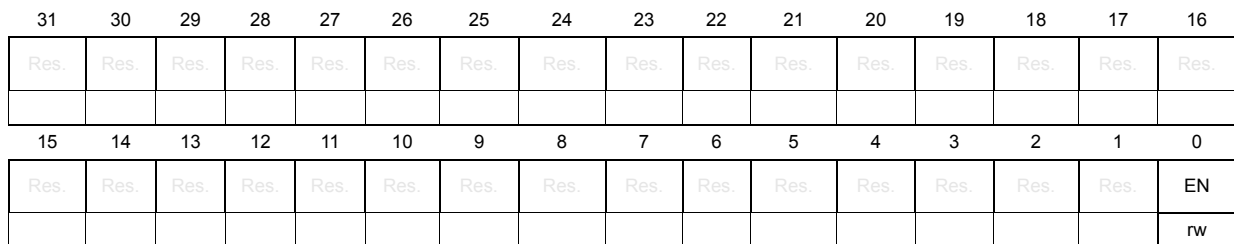


Bits 31: 0 **VERSION**: Version of the DSI Host  
 This RO register contains the version of the DSI Host

### 29.15.2 DSI Host Control Register (DSI\_CR)

Address offset: 0x0004

Reset value: 0x0000 0000



Bits 31: 1 Reserved, must be kept at reset value  
 Bit 0 **EN**: Enable  
 This bit configures the DSI Host in either power-up mode or to reset.  
 0: DSI Host is disabled (under reset).  
 1: DSI Host is enabled.

### 29.15.3 DSI HOST Clock Control Register (DSI\_CCR)

Address offset: 0x0008

Reset value: 0x0000 0000

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

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

Bits 15: 8 **TOCKDIV**: Timeout Clock Division

This field indicates the division factor for the Time Out clock used as the timing unit in the configuration of HS to LP and LP to HS transition error.

Bits 7: 0 **TXECKDIV**: TX Escape Clock Division

This field indicates the division factor for the TX Escape clock source (lanebyteclk). The values 0 and 1 stop the TX\_ESC clock generation.

### 29.15.4 DSI Host LTDC VCID Register (DSI\_LVCIDR)

Address offset: 0x000C

Reset value: 0x0000 0000

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

Bits 31: 2 Reserved, must be kept at reset value

Bits 1: 0 **VCID**: Virtual Channel ID

These bits configure the virtual channel ID for the LTDC interface traffic.

### 29.15.5 DSI Host LTDC Color Coding Register (DSI\_LCOLCR)

Address offset: 0x0010

Reset value: 0x0000 0000

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

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

Bit 8 **LPE**: Loosely Packet Enable

This bit enables the loosely packed variant to 18-bit configuration

0: Loosely Packet variant disabled

1: Loosely Packet variant enabled

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

Bit 3: 0 **COLC**: Color Coding

This field configures the DPI color coding

0000: 16-bit configuration 1

0001: 16-bit configuration 2

0010: 16-bit configuration 3

0011: 18-bit configuration 1

0100: 18-bit configuration 2

0101: 24-bit

0110-1111: Reserved

### 29.15.6 DSI Host LTDC Polarity Configuration Register (DSI\_LPCR)

Address offset: 0x0014

Reset value: 0x0000 0000

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

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

Bit 2 **HSP**: HSYNC Polarity

This bit configures the polarity of HSYNC pin.

0: HSYNC pin active high (default).

1: VSYNC pin active low.



Bit 1 **VSP**: VSYNC Polarity  
 This bit configures the polarity of VSYNC pin.  
 0: Shutdown pin active high (default).  
 1: Shutdown pin active low.

Bit 0 **DEP**: Data Enable Polarity  
 This bit configures the polarity of Data Enable pin.  
 0: Data Enable pin active high (default).  
 1: Data Enable pin active low.

### 29.15.7 DSI Host Low-Power mode Configuration Register (DSI\_LPMCR)

Address offset: 0x0018

Reset value: 0x0000 0000

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

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

Bits 23: 16 **LPSIZE**: Largest Packet Size  
 This field is used for the transmission of commands in Low-Power mode. It defines the size, in bytes, of the largest packet that can fit in a line during VSA, VBP and VFP regions.

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

Bits 7: 0 **VLPSIZE**: VACT Largest Packet Size  
 This field is used for the transmission of commands in Low-Power mode. It defines the size, in bytes, of the largest packet that can fit in a line during VACT regions.

### 29.15.8 DSI Host Protocol Configuration Register (DSI\_PCR)

Address offset: 0x002C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	CRCRXE	ECCRXE	BTAE	ETRXE	ETTXE
											rw	rw	rw	rw	rw

Bits 31: 5 Reserved, must be kept at reset value

Bit 4 **CRCRXE**: CRC Reception Enable

This bit enables the CRC reception and error reporting.

0: CRC reception is disabled.

1: CRC reception is enabled.

Bit 3 **ECRCXE**: ECC Reception Enable

This bit enables the ECC reception, error correction, and reporting.

0: ECC reception is disabled.

1: ECC reception is enabled.

Bit 2 **BTAE**: Bus Turn Around Enable

This bit enables the Bus Turn-Around (BTA) request.

0: Bus Turn-Around request is disabled.

1: Bus Turn-Around request is enabled.

Bit 1 **ETRXE**: EoTp Reception Enable

This bit enables the EoTp reception.

0: EoTp reception is disabled.

1: EoTp reception is enabled.

Bit 0 **ETTXE**: EoTp Transmission Enable

This bit enables the EoTP transmission.

0: EoTp transmission is disabled.

1: EoTp transmission is enabled.

### 29.15.9 DSI Host Generic VCID Register (DSI\_GVCIDR)

Address offset: 0x0030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	VCID[1:0]

Bits 31: 2 Reserved, must be kept at reset value

Bits 1: 0 **VCID**: Virtual Channel ID

This field indicates the Generic interface read-back virtual channel identification.

### 29.15.10 DSI Host mode Configuration Register (DSI\_MCR)

Address offset: 0x0034

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDM
															rw

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

Bit 0 **CMDM**: Command mode

This bit configures the DSI Host in either Video or Command mode.

0: DSI Host is configured in video mode.

1: DSI Host is configured in Command mode.

### 29.15.11 DSI Host video mode Configuration Register (DSI\_VMCR)

Address offset: 0x0038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PGO	Res.	Res.	Res.	PGM	Res.	Res.	Res.	PGE
							rw				rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPCE	FBTAAE	LPHFE	LPHBPE	LPVAE	LPVFPE	LPVBPE	LPVSAE	Res.	Res.	Res.	Res.	Res.	Res.	VMT[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw							rw	

Bits 31: 25 Reserved, must be kept at reset value

Bit 24 **PGO**: Pattern Generator Orientation

This bit configures the color bar orientation.

0: Vertical color bars.

1: Horizontal color bars.

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

Bit 20 **PGM**: Pattern Generator mode

This bit configures the pattern generator mode.

0: Color bars (horizontal or vertical).

1: BER pattern (vertical only).

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

- Bit 16 **PGE**: Pattern Generator Enable  
This bit enables the video mode pattern generator.  
0: Pattern generator is disabled.  
1: Pattern generator is enabled.
- Bit 15 **LPCE**: Low-Power Command Enable  
This bit enables the command transmission only in Low-Power mode.  
0: Command transmission in Low-Power mode is disabled.  
1: Command transmission in Low-Power mode is enabled.
- Bit 14 **FBTAAE**: Frame Bus-Turn-Around Acknowledge Enable  
This bit enables the request for an acknowledge response at the end of a frame.  
0: Acknowledge response at the end of a frame is disabled.  
1: Acknowledge response at the end of a frame is enabled.
- Bit 13 **LPHFPE**: Low-Power Horizontal Front-Porch Enable  
This bit enables the return to Low-Power inside the Horizontal Front Porch (HFP) period when timing allows.  
0: Return to Low-Power inside the HFP period is disabled.  
1: Return to Low-Power inside the HFP period is enabled.
- Bit 12 **LPHBPE**: Low-Power Horizontal Back-Porch Enable  
This bit enables the return to Low-Power inside the Horizontal Back Porch (HBP) period when timing allows.  
0: Return to Low-Power inside the HBP period is disabled.  
1: Return to Low-Power inside the HBP period is enabled.
- Bit 11 **LPVAE**: Low-Power Vertical Active Enable  
This bit enables to return to Low-Power inside the Vertical Active (VACT) period when timing allows.  
0: Return to Low-Power inside the VACT is disabled.  
1: Return to Low-Power inside the VACT is enabled.
- Bit 10 **LPVFPE**: Low-power Vertical Front-porch Enable  
This bit enables to return to Low-Power inside the Vertical Front Porch (VFP) period when timing allows.  
0: Return to Low-Power inside the VFP is disabled.  
1: Return to Low-Power inside the VFP is enabled.
- Bit 9 **LPVBPE**: Low-power Vertical Back-Porch Enable  
This bit enables to return to Low-Power inside the Vertical Back Porch (VBP) period when timing allows.  
0: Return to Low-Power inside the VBP is disabled.  
1: Return to Low-Power inside the VBP is enabled.

Bit 8 **LPVSAE**: Low-Power Vertical Sync Active Enable

This bit enables to return to Low-Power inside the Vertical Sync time (VSA) period when timing allows.

0: Return to Low-Power inside the VSA is disabled.

1: Return to Low-Power inside the VSA is enabled

Bits 7: 2 Reserved, must be kept at reset value

Bits 1: 0 **VMT**: video mode Type

This field configures the video mode transmission type :

00: Non-Burst with sync pulses.

01: Non-Burst with sync events.

1x: Burst mode

### 29.15.12 DSI Host video Packet Configuration Register (DSI\_VPCR)

Address offset: 0x003C

Reset value: 0x0000 0000

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

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

Bits 13: 0 **VPSIZE**: video Packet Size

This field configures the number of pixels in a single video packet.

For 18-bit not loosely packed data types, this number must be a multiple of 4.

For YCbCr data types, it must be a multiple of 2 as described in the DSI specification.

### 29.15.13 DSI Host video Chunks Configuration Register (DSI\_VCCR)

Address offset: 0x0040

Reset value: 0x0000 0000

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

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

Bits 12: 0 **NUMC**: Number of Chunks

This register configures the number of chunks to be transmitted during a Line period (a chunk consists of a video packet and a null packet).

If set to 0 or 1, the video line is transmitted in a single packet.

If set to 1, the packet is part of a chunk, so a null packet follows it if NPSIZE > 0.

Otherwise, multiple chunks are used to transmit each video line.

### 29.15.14 DSI Host Video Null Packet Configuration Register (DSI\_VNPCR)

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

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

Bits 12: 0 **NPSIZE**: Null Packet Size

This field configures the number of bytes inside a null packet.

Setting to 0 disables the null packets.

### 29.15.15 DSI Host Video HSA Configuration Register (DSI\_VHSACR)

Address offset: 0x0048

Reset value: 0x0000 0000

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

Bits 31: 12 Reserved, must be kept at reset value

Bits 11: 0 **HSA**: Horizontal Synchronism Active duration

This fields configures the Horizontal Synchronism Active period in lane byte clock cycles..

### 29.15.16 DSI Host Video HBP Configuration Register (DSI\_VHBPCR)

Address offset: 0x004C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	HBP[11:0]											
rw															

Bits 31: 12 Reserved, must be kept at reset value

Bits 11: 0 **HBP**: Horizontal Back-Porch duration

This fields configures the Horizontal Back-Porch period in lane byte clock cycles.

### 29.15.17 DSI Host Video Line Configuration Register (DSI\_VLCR)

Address offset: 0x0050

Reset value: 0x0000 0000

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

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

Bits 14: 0 **HLINE**: Horizontal Line duration

This fields configures the total of the Horizontal Line period (HSA+HBP+HACT+HFP) counted in lane byte clock cycles.

### 29.15.18 DSI Host Video VSA Configuration Register (DSI\_VVSACR)

Address offset: 0x0054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	VSA[9:0]									
rw															



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

Bits 9: 0 **VSA**: Vertical Synchronism Active duration

This fields configures the Vertical Synchronism Active period measured in number of horizontal lines.

### 29.15.19 DSI Host Video VBP Configuration Register (DSI\_VVBPCR)

Address offset: 0x0058

Reset value: 0x0000 0000

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

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

Bits 9: 0 **VBP**: Vertical Back-Porch duration

This fields configures the Vertical Back-Porch period measured in number of horizontal lines.

### 29.15.20 DSI Host Video VFP Configuration Register (DSI\_VVFPCR)

Address offset: 0x005C

Reset value: 0x0000 0000

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

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

Bits 9: 0 **VFP**: Vertical Front-Porch duration

This fields configures the Vertical Front-Porch period measured in number of horizontal lines.



**29.15.21 DSI Host Video VA Configuration Register (DSI\_VVACR)**

Address offset: 0x0060

Reset value: 0x0000 0000

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

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

Bits 13: 0 **VA**: Vertical Active duration

This fields configures the Vertical Active period measured in number of horizontal lines.

**29.15.22 DSI Host LTDC Command Configuration Register (DSI\_LCCR)**

Address offset: 0x0064

Reset value: 0x0000 0000

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

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

Bits 15: 0 **CMDSIZE**: Command Size

This field configures the maximum allowed size for an LTDC write memory command, measured in pixels. Automatic partitioning of data obtained from LTDC is permanently enabled.

### 29.15.23 DSI Host Command mode Configuration Register (DSI\_CMCR)

Address offset: 0x0068

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MRDPS	Res.	Res.	Res.	Res.	DLWTX	DSR0TX	DSW1TX	DSW0TX
							rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GLWTX	GSR 2TX	GSR 1TX	GSR 0TX	GSW 2TX	GSW 1TX	GSW 0TX	Res.	Res.	Res.	Res.	Res.	Res.	ARE	TEARE
	rw	rw	rw	rw	rw	rw	rw							rw	rw

Bits 31: 25 Reserved, must be kept at reset value

Bit 24 **MRDPS**: Maximum Read Packet Size

This bit configures the maximum read packet size command transmission type:

0: High-speed.

1: Low-power.

Bits 23: 20 Reserved, must be kept at reset value

Bit 19 **DLWTX**: DCS Long Write Transmission

This bit configures the DCS long write packet command transmission type:

0: High-speed.

1: Low-power.

Bit 18 **DSR0TX**: DCS Short Read Zero parameter Transmission

This bit configures the DCS short read packet with zero parameter command transmission type:

0: High-speed.

1: Low-power.

Bit 17 **DSW1TX**: DCS Short Read One parameter Transmission

This bit configures the DCS short read packet with one parameter command transmission type:

0: High-speed.

1: Low-power.

Bit 16 **DSW0TX**: DCS Short Write Zero parameter Transmission

This bit configures the DCS short write packet with zero parameter command transmission type:

0: High-speed.

1: Low-power.

Bit 15 Reserved, must be kept at reset value

Bit 14 **GLWTX**: Generic Long Write Transmission

This bit configures the Generic long write packet command transmission type :

0: High-speed.

1: Low-power.

- Bit 13 **GSR2TX**: Generic Short Read Two parameters Transmission  
This bit configures the Generic short read packet with two parameters command transmission type:  
0: High-speed.  
1: Low-power.
- Bit 12 **GSR1TX**: Generic Short Read One parameters Transmission  
This bit configures the Generic short read packet with one parameters command transmission type:  
0: High-speed.  
1: Low-power.
- Bit 11 **GSR0TX**: Generic Short Read Zero parameters Transmission  
This bit configures the Generic short read packet with zero parameters command transmission type:  
0: High-speed.  
1: Low-power.
- Bit 10 **GSW2TX**: Generic Short Write Two parameters Transmission  
This bit configures the Generic short write packet with two parameters command transmission type:  
0: High-speed.  
1: Low-power.
- Bit 9 **GSW1TX**: Generic Short Write One parameters Transmission  
This bit configures the Generic short write packet with one parameters command transmission type:  
0: High-speed.  
1: Low-power.
- Bit 8 **GSW0TX**: Generic Short Write Zero parameters Transmission  
This bit configures the Generic short write packet with zero parameters command transmission type:  
0: High-speed.  
1: Low-power.
- Bits 7: 2 Reserved, must be kept at reset value
- Bit 1 **ARE**: Acknowledge Request Enable  
This bit enables the acknowledge request after each packet transmission:  
0: Acknowledge request is disabled.  
1: Acknowledge request is enabled.
- Bit 0 **TEARE**: Tearing Effect Acknowledge Request Enable  
This bit enables the tearing effect acknowledge request:  
0: Tearing effect acknowledge request is disabled.  
1: Tearing effect acknowledge request is enabled.

### 29.15.24 DSI Host Generic Header Configuration Register (DSI\_GHCR)

Address offset: 0x006C

Reset value: 0x0000 0000

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

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

Bits 23: 16 **WCMSB**: WordCount MSB

This field configures the most significant byte of the header packet’s word count for long packets or data 1 for short packet.

Bits 15: 8 **WCLSB**: WordCount LSB

This field configures the most significant byte of the header packet’s word count for long packets or data 1 for short packet.

Bits 7: 6 **VCID**: Channel

This field configures the virtual channel ID of the header packet.

Bits 5: 0 **DT**: Type

This field configures the packet data type of the header packet.

### 29.15.25 DSI Host Generic Payload Data Register (DSI\_GPDR)

Address offset: 0x0070

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA4[7:0]								DATA3[7:0]							
rw								rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA2[7:0]								DATA1[7:0]							
rw								rw							

Bits 31: 24 **DATA4**: Payload Byte 4

This field indicates the byte 4 of the packet payload.

Bits 23: 16 **DATA3**: Payload Byte 3

This field indicates the byte 3 of the packet payload.

Bits 15: 8 **DATA2**: Payload Byte 2

This field indicates the byte 2 of the packet payload.

Bits 7: 0 **DATA1**: Payload Byte 1

This field indicates the byte 1 of the packet payload.

### 29.15.26 DSI Host Generic Packet Status Register (DSI\_GPSR)

Address offset: 0x0074

Reset value: 0x0000 0015

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	RCB	PRDFF	PRDFE	PWRFF	PWRFE	CMDFF	CMDFE
									ro	ro	ro	ro	ro	ro	ro

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

Bit 6 **RCB**: Read Command Busy

This bit is set when a read command is issued and cleared when the entire response is stored in the FIFO:

0: No read command on going.

1: Read command on going.

Bit 5 **PRDFF**: Payload Read FIFO Full

This bit indicates the full status of the generic read payload FIFO:

0: Read payload FIFO not full.

1: Read payload FIFO full.

Bit 4 **PRDFE**: Payload Read FIFO Empty

This bit indicates the empty status of the generic read payload FIFO:

0: Read payload FIFO not empty.

1: Read payload FIFO empty.

Bit 3 **PWRFF**: Payload Write FIFO Full

This bit indicates the full status of the generic write payload FIFO:

0: Write payload FIFO not full.

1: Write payload FIFO full.

Bit 2 **PWRFE**: Payload Write FIFO Empty

This bit indicates the empty status of the generic write payload FIFO:

0: Write payload FIFO not empty.

1: Write payload FIFO empty.

Bit 1 **CMDFF**: Command FIFO Full

This bit indicates the full status of the generic command FIFO:

0: Write payload FIFO not full.

1: Write payload FIFO full.

Bit 0 **CMDFE**: Command FIFO Empty

This bit indicates the empty status of the generic command FIFO:

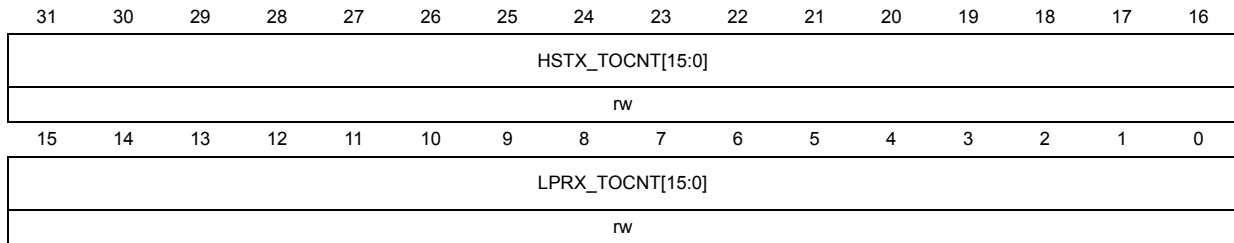
0: Write payload FIFO not empty.

1: Write payload FIFO empty.

### 29.15.27 DSI Host Timeout Counter Configuration Register 0 (DSI\_TCCR0)

Address offset: 0x0078

Reset value: 0x0000 0000



Bits 31: 16 **HSTX\_TOCNT**: High-Speed Transmission Timeout Counter

This field configures the timeout counter that triggers a High-Speed transmission timeout contention detection (measured in TOCKDIV cycles).

If using the non-burst mode and there is no sufficient time to switch from High-Speed to Low-Power and back in the period which is from one line data finishing to the next line sync start, the DSI link returns the Low-Power state once per frame, then you should configure the TOCKDIV and HSTX\_TOCNT to be in accordance with:

$$HSTX\_TOCNT * lanebyteclkperiod * TOCKDIV \geq \text{the time of one FRAME data transmission} * (1 + 10\%)$$

In burst mode, RGB pixel packets are time-compressed, leaving more time during a scan line. Therefore, if in burst mode and there is sufficient time to switch from High-Speed to Low-Power and back in the period of time from one line data finishing to the next line sync start, the DSI link can return Low-Power mode and back in this time interval to save power. For this, configure the TOCKDIV and HSTX\_TOCNT to be in accordance with:

$$HSTX\_TOCNT * lanebyteclkperiod * TOCKDIV \geq \text{the time of one LINE data transmission} * (1 + 10\%)$$

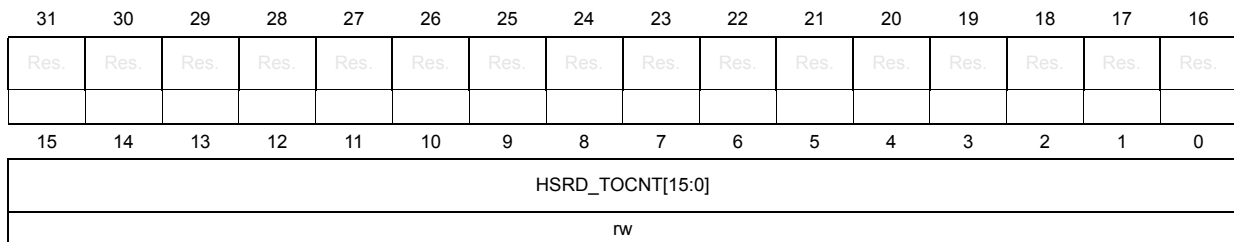
Bits 15: 0 **LPRX\_TOCNT**: Low-power Reception Timeout Counter

This field configures the timeout counter that triggers a Low-Power reception timeout contention detection (measured in TOCKDIV cycles).

### 29.15.28 DSI Host Timeout Counter Configuration Register 1 (DSI\_TCCR1)

Address offset: 0x007C

Reset value: 0x0000 0000



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

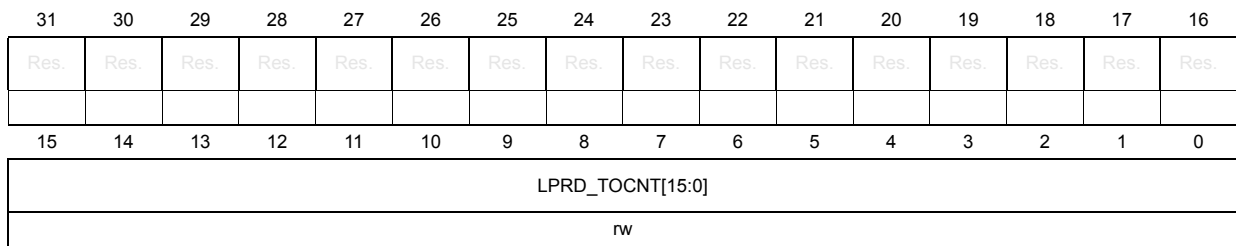
Bits 15: 0 **HSRD\_TOCNT**: High-Speed Read Timeout Counter

This field sets a period for which the DSI Host keeps the link still, after sending a High-Speed read operation. This period is measured in cycles of lanebyteclk. The counting starts when the D-PHY enters the Stop state and causes no interrupts.

### 29.15.29 DSI Host Timeout Counter Configuration Register 2 (DSI\_TCCR2)

Address offset: 0x0080

Reset value: 0x0000 0000



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

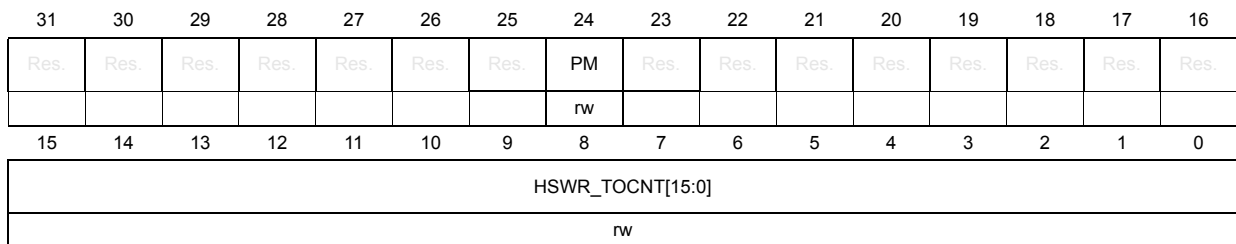
Bits 15: 0 **LPRD\_TOCNT[15:0]**: Low-Power Read Timeout Counter

This field sets a period for which the DSI Host keeps the link still, after sending a Low-Power read operation. This period is measured in cycles of lanebyteclk. The counting starts when the D-PHY enters the Stop state and causes no interrupts.

### 29.15.30 DSI Host Timeout Counter Configuration Register 3 (DSI\_TCCR3)

Address offset: 0x0084

Reset value: 0x0000 0000



Bits 31: 25 Reserved, must be kept at reset value

Bit 24 **PM**: Presp mode

When set to 1, this bit ensures that the peripheral response timeout caused by HSWR\_TOCNT is used only once per LTDC frame in Command mode, when both the following conditions are met:

- dpivsync\_edpiwms has risen and fallen.
- Packets originated from LTDC in Command mode have been transmitted and its FIFO is empty again.

In this scenario no non-LTDC command requests are sent to the D-PHY, even if there is traffic from generic interface ready to be sent, making it return to stop state. When it does so, PRES\_P\_TO counter is activated and only when it finishes does the controller send any other traffic that is ready.

Bits 23: 16 Reserved, must be kept at reset value

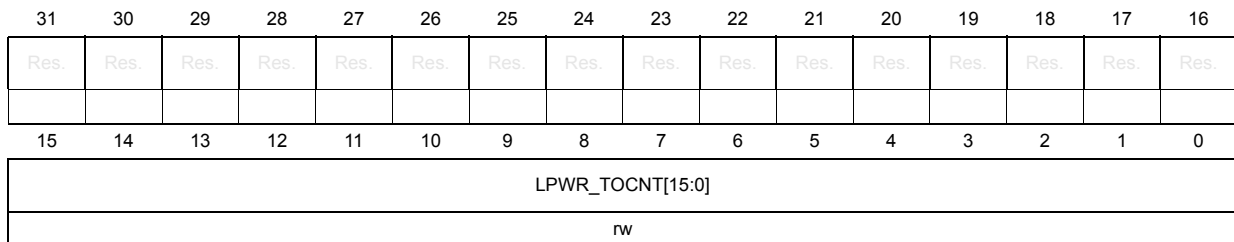
Bits 15: 0 **HSWR\_TOCNT**: High-Speed Write Timeout Counter

This field sets a period for which the DSI Host keeps the link inactive after sending a High-Speed write operation. This period is measured in cycles of lanebyteclk. The counting starts when the D-PHY enters the Stop state and causes no interrupts.

### 29.15.31 DSI Host Timeout Counter Configuration Register 4 (DSI\_TCCR4)

Address offset: 0x0088

Reset value: 0x0000 0000



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

Bits 15: 0 **LSWR\_TOCNT**: Low-Power Write Timeout Counter

This field sets a period for which the DSI Host keeps the link still, after sending a Low-Power write operation. This period is measured in cycles of lanebyteclk. The counting starts when the D-PHY enters the Stop state and causes no interrupts.



### 29.15.32 DSI Host Timeout Counter Configuration Register 5 (DSI\_TCCR5)

Address offset: 0x008C

Reset value: 0x0000 0000

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

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

Bits 15: 0 **BTA\_TOCNT**: Bus-Turn-Around Timeout Counter

This field sets a period for which the DSI Host keeps the link still, after completing a Bus-Turn-Around. This period is measured in cycles of lanebyteclk. The counting starts when the D-PHY enters the Stop state and causes no interrupts.

### 29.15.33 DSI Host Clock Lane Configuration Register (DSI\_CLCR)

Address offset: 0x0094

Reset value: 0x0000 0000

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

Bits 31: 2 Reserved, must be kept at reset value

Bit 1 **ACR**: Automatic Clock lane Control

This bit enables the automatic mechanism to stop providing clock in the clock lane when time allows.

0: Automatic Clock Lane control disabled

1: Automatic Clock Lane control enabled

Bit 0 **DPCC**: D-PHY Clock Control

This bit controls the D-PHY Clock state:

0: Clock Lane is in Low-Power mode

1: Clock Lane is running in High-Speed mode

### 29.15.34 DSI Host Clock Lane Timer Configuration Register (DSI\_CLTCR)

Address offset: 0x0098

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	HS2LP_TIME[9:0]									
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	LP2HS_TIME[9:0]									
rw															

Bits 31: 26 Reserved, must be kept at reset value

Bits 25: 16 **HS2LP\_TIME**: High-Speed to Low-Power Time

This field configures the maximum time that the D-PHY clock lane takes to go from High-Speed to Low-Power transmission measured in lane byte clock cycles.

Bits 15: 10 Reserved, must be kept at reset value

Bits 9: 0 **LP2HS\_TIME**: Low-Power to High-Speed Time

This field configures the maximum time that the D-PHY clock lane takes to go from Low-Power to High-Speed transmission measured in lane byte clock cycles.

### 29.15.35 DSI Host Data Lane Timer Configuration Register (DSI\_DLTCR)

Address offset: 0x009C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HS2LP_TIME[7:0]							LP2HS_TIME[7:0]								
rw							rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MRD_TIME[14:0]														
rw															

Bits 31: 24 **HS2LP\_TIME**: High-Speed To Low-Power Time

This field configures the maximum time that the D-PHY data lanes take to go from High-Speed to Low-Power transmission measured in lane byte clock cycles.

Bits 23: 16 **LP2HS\_TIME**: Low-Power To High-Speed Time

This field configures the maximum time that the D-PHY data lanes take to go from Low-Power to High-Speed transmission measured in lane byte clock cycles.

Bit 15 Reserved, must be kept at reset value

Bits 14: 0 **MRD\_TIME**: Maximum Read Time

This field configures the maximum time required to perform a read command in lane byte clock cycles. This register can only be modified when no read command is in progress.

### 29.15.36 DSI Host PHY Control Register (DSI\_PCTLR)

Address offset: 0x00A0

Reset value: 0x0000 0000

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

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

Bit 2 **CKE**: Clock Enable

This bit enables the D-PHY Clock Lane module:

0: D-PHY Clock lane module is disabled.

1: D-PHY Clock lane module is enabled.

Bit 1 **DEN**: Digital Enable

When set to 0, this bit places the digital section of the D-PHY in the reset state

0: The digital section of the D-PHY is in the reset state.

1: The digital section of the D-PHY is enabled.

Bit 0 Reserved

### 29.15.37 DSI Host PHY Configuration Register (DSI\_PCONFR)

Address offset: 0x00A4

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
SW_TIME[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	NL[1:0]	
rw														rw	

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

Bits 15: 8 **SW\_TIME**: Stop Wait Time

This field configures the minimum wait period to request a High-Speed transmission after the Stop state.

Bits 7: 2 Reserved, must be kept at reset value

Bits 1: 0 **NL**: Number of Lanes

This field configures the number of active data lanes:

00: One data lane (lane 0)

01: Two data lanes (lanes 0 and 1) - Reset value

10: Reserved

11: Reserved

### 29.15.38 DSI Host PHY ULPS Control Register (DSI\_PUCR)

Address offset: 0x00A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	UEDL	URDL	UECL	URCL
												rw	rw	rw	rw

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

Bit 3 **UEDL**: ULPS Exit on Data Lane

ULPS mode Exit on all active data lanes.

0: No exit request

1: Exit ULPS mode on all active data lane URDL

Bit 2 **URDL**: ULPS Request on Data Lane

ULPS mode Request on all active data lanes.

0: No ULPS request

1: Request ULPS mode on all active data lane UECL

Bit 1 **UECL**: ULPS Exit on Clock Lane

ULPS mode Exit on clock lane.

0: No exit request

1: Exit ULPS mode on clock lane

Bit 0 **URCL**: ULPS Request on Clock Lane

ULPS mode Request on clock lane.

0: No ULPS request

1: Request ULPS mode on clock lane

### 29.15.39 DSI Host PHY TX Triggers Configuration Register (DSI\_PTTCR)

Address offset: 0x00AC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	TX_TRIG[3:0]			
												rw			

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

Bits 3: 0 **TX\_TRIG**: Transmission Trigger

Escape mode Transmit Trigger 0-3.

Only one bit of TX\_TRIG is asserted at any given time.

### 29.15.40 DSI Host PHY Status Register (DSI\_PSR)

Address offset: 0x00B0

Reset value: 0x0000 1528

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	UAN1	PSS1	RUE0	UAN0	PSS0	UANC	PSSC	PD	Res.
							ro	ro	ro	ro	ro	ro	ro	ro	

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

Bit 8 **UAN1**: ULPS Active Not lane 1

This bit indicates the status of ulpsactivenot1lane D-PHY signal.

Bit 7 **PSS1**: PHY Stop State lane 1

This bit indicates the status of phystopstate1lane D-PHY signal.

Bit 6 **RUE0**: RX ULPS Escape lane 0

This bit indicates the status of rxulpsesc0lane D-PHY signal.

Bit 5 **UAN0**: ULPS Active Not lane 1

This bit indicates the status of ulpsactivenot0lane D-PHY signal.

Bit 4 **PSS0**: PHY Stop State lane 0

This bit indicates the status of phystopstate0lane D-PHY signal.

Bit 3 **UANC**: ULPS Active Not Clock lane

This bit indicates the status of ulpsactivenotcklane D-PHY signal.

- Bit 2 **PSSC**: PHY Stop State Clock lane  
This bit indicates the status of phystopstatecklane D-PHY signal.
- Bit 1 **PD**: PHY Direction  
This bit indicates the status of phydirection D-PHY signal.
- Bit 0 Reserved

### 29.15.41 DSI Host Interrupt & Status Register 0 (DSI\_ISR0)

Address offset: 0x00BC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PE4	PE3	PE2	PE1	PE0
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AE15	AE14	AE13	AE12	AE11	AE10	AE9	AE8	AE7	AE6	AE5	AE4	AE3	AE2	AE1	AE0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

- Bit 20 **PE4**: PHY Error 4  
This bit indicates the LP1 contention error ErrContentionLP1 from lane 0.
- Bit 19 **PE3**: PHY Error 3  
This bit indicates the LP0 contention error ErrContentionLP0 from lane 0.
- Bit 18 **PE2**: PHY Error 2  
This bit indicates the ErrControl error from Lane 0.
- Bit 17 **PE1**: PHY Error 1  
This bit indicates the ErrSyncEsc Low-Power transmission synchronization error from lane 0.
- Bit 16 **PE0**: PHY Error 0  
This bit indicates the ErrEsc escape entry error from lane 0.
- Bit 15 **AE15**: Acknowledge Error 15  
This bit retrieves the DSI protocol violation from the Acknowledge error report.
- Bit 14 **AE14**: Acknowledge Error 14  
This bit retrieves the reserved (specific to the device) from the Acknowledge error report.
- Bit 13 **AE13**: Acknowledge Error 13  
This bit retrieves the invalid transmission length from the Acknowledge error report.
- Bit 12 **AE12**: Acknowledge Error 12  
This bit retrieves the DSI VC ID Invalid from the Acknowledge error report.
- Bit 11 **AE11**: Acknowledge Error 11  
This bit retrieves the not recognized DSI data type from the Acknowledge error report.
- Bit 10 **AE10**: Acknowledge Error 10  
This bit retrieves the checksum error (long packet only) from the Acknowledge error report.

- Bit 9 **AE9**: Acknowledge Error 9  
This bit retrieves the ECC error, multi-bit (detected, not corrected) from the Acknowledge error report.
- Bit 8 **AE8**: Acknowledge Error 8  
This bit retrieves the ECC error, single-bit (detected and corrected) from the Acknowledge error report.
- Bit 7 **AE7**: Acknowledge Error 7  
This bit retrieves the reserved (specific to the device) from the Acknowledge error report.
- Bit 6 **AE6**: Acknowledge Error 6  
This bit retrieves the False Control error from the Acknowledge error report.
- Bit 5 **AE5**: Acknowledge Error 5  
This bit retrieves the Peripheral Timeout error from the Acknowledge error report.
- Bit 4 **AE4**: Acknowledge Error 4  
This bit retrieves the LP Transmit Sync error from the Acknowledge error report.
- Bit 3 **AE3**: Acknowledge Error 3  
This bit retrieves the Escape mode Entry Command error from the Acknowledge error report.
- Bit 2 **AE2**: Acknowledge Error 2  
This bit retrieves the EoT Sync error from the Acknowledge error report.
- Bit 1 **AE1**: Acknowledge Error 1  
This bit retrieves the SoT Sync error from the Acknowledge error report.
- Bit 0 **AE0**: Acknowledge Error 0  
This bit retrieves the SoT error from the Acknowledge error report.

### 29.15.42 DSI Host Interrupt & Status Register 1 (DSI\_ISR1)

Address offset: 0x00C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	GPRXE	GPRDE	GPTXE	GPWRE	GCWRE	LPWRE	EOTPE	PSE	CRCE	ECCME	ECCSE	TOLPRX	TOHSTX
			r	r	r	r	r	r	r	r	r	r	r	r	r

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

- Bit 12 **GPRXE**: Generic Payload Receive Error  
This bit indicates that during a generic interface packet read back, the payload FIFO becomes full and the received data is corrupted.
- Bit 11 **GPRDE**: Generic Payload Read Error  
This bit indicates that during a DCS read data, the payload FIFO becomes empty and the data sent to the interface is corrupted.



- Bit 10 **GPTXE**: Generic Payload Transmit Error  
This bit indicates that during a Generic interface packet build, the payload FIFO becomes empty and corrupt data is sent.
- Bit 9 **GPWRE**: Generic Payload Write Error  
This bit indicates that the system tried to write a payload data through the Generic interface and the FIFO is full. Therefore, the payload is not written.
- Bit 8 **GCWRE**: Generic Command Write Error  
This bit indicates that the system tried to write a command through the Generic interface and the FIFO is full. Therefore, the command is not written.
- Bit 7 **LPWRE**: LTDC Payload Write Error  
This bit indicates that during a DPI pixel line storage, the payload FIFO becomes full and the data stored is corrupted.
- Bit 6 **EOTPE**: EoTp Error  
This bit indicates that the EoTp packet is not received at the end of the incoming peripheral transmission.
- Bit 5 **PSE**: Packet Size Error  
This bit indicates that the packet size error is detected during the packet reception.
- Bit 4 **CRCE**: CRC Error  
This bit indicates that the CRC error is detected in the received packet payload.
- Bit 3 **ECCME**: ECC Multi-bit Error  
This bit indicates that the ECC multiple error is detected in a received packet.
- Bit 2 **ECCSE**: ECC Single-bit Error  
This bit indicates that the ECC single error is detected and corrected in a received packet.
- Bit 1 **TOLPRX**: Timeout Low-Power Reception  
This bit indicates that the Low-Power reception timeout counter reached the end and contention is detected.
- Bit 0 **TOHSTX**: Timeout High-Speed Transmission  
This bit indicates that the High-Speed transmission timeout counter reached the end and contention is detected.

### 29.15.43 DSI Host Interrupt Enable Register 0 (DSI\_IER0)

Address offset: 0x00C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PE4IE	PE3IE	PE2IE	PE1IE	PE0IE
											rW	rW	rW	rW	rW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AE15IE	AE14IE	AE13IE	AE12IE	AE11IE	AE10IE	AE9IE	AE8IE	AE7IE	AE6IE	AE5IE	AE4IE	AE3IE	AE2IE	AE1IE	AE0IE
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW



- Bits 31: 21 Reserved, must be kept at reset value
- Bit 20 **PE4IE**: PHY Error 4 Interrupt Enable  
This bit enables the interrupt generation on PHY Error 4.  
0: Interrupt on PHY Error 4 disabled.  
1: Interrupt on PHY Error 4 enabled.
- Bit 19 **PE3IE**: PHY Error 3 Interrupt Enable  
This bit enables the interrupt generation on PHY Error 4.  
0: Interrupt on PHY Error 3 disabled.  
1: Interrupt on PHY Error 3 enabled.
- Bit 18 **PE2IE**: PHY Error 2 Interrupt Enable  
This bit enables the interrupt generation on PHY Error 2.  
0: Interrupt on PHY Error 2 disabled.  
1: Interrupt on PHY Error 2 enabled.
- Bit 17 **PE1IE**: PHY Error 1 Interrupt Enable  
This bit enables the interrupt generation on PHY Error 1.  
0: Interrupt on PHY Error 1 disabled.  
1: Interrupt on PHY Error 1 enabled.
- Bit 16 **PE0IE**: PHY Error 0 Interrupt Enable  
This bit enables the interrupt generation on PHY Error 0.  
0: Interrupt on PHY Error 0 disabled.  
1: Interrupt on PHY Error 0 enabled.
- Bit 15 **AE15IE**: Acknowledge Error 15 Interrupt Enable  
This bit enables the interrupt generation on Acknowledge Error 15.  
0: Interrupt on Acknowledge Error 15 disabled.  
1: Interrupt on Acknowledge Error 15 enabled.
- Bit 14 **AE14IE**: Acknowledge Error 14 Interrupt Enable  
This bit enables the interrupt generation on Acknowledge Error 14.  
0: Interrupt on Acknowledge Error 14 disabled.  
1: Interrupt on Acknowledge Error 14 enabled.
- Bit 13 **AE13IE**: Acknowledge Error 13 Interrupt Enable  
This bit enables the interrupt generation on Acknowledge Error 13.  
0: Interrupt on Acknowledge Error 13 disabled.  
1: Interrupt on Acknowledge Error 13 enabled.
- Bit 12 **AE12IE**: Acknowledge Error 12 Interrupt Enable  
This bit enables the interrupt generation on Acknowledge Error 12.  
0: Interrupt on Acknowledge Error 12 disabled.  
1: Interrupt on Acknowledge Error 12 enabled.
- Bit 11 **AE11IE**: Acknowledge Error 11 Interrupt Enable  
This bit enables the interrupt generation on Acknowledge Error 11.  
0: Interrupt on Acknowledge Error 11 disabled.  
1: Interrupt on Acknowledge Error 11 enabled.

- Bit 10 **AE10IE**: Acknowledge Error 10 Interrupt Enable  
This bit enables the interrupt generation on Acknowledge Error 10.  
0: Interrupt on Acknowledge Error 10 disabled.  
1: Interrupt on Acknowledge Error 10 enabled.
- Bit 9 **AE9IE**: Acknowledge Error 9 Interrupt Enable  
This bit enables the interrupt generation on Acknowledge Error 9.  
0: Interrupt on Acknowledge Error 9 disabled.  
1: Interrupt on Acknowledge Error 9 enabled.
- Bit 8 **AE8IE**: Acknowledge Error 8 Interrupt Enable  
This bit enables the interrupt generation on Acknowledge Error 8.  
0: Interrupt on Acknowledge Error 8 disabled.  
1: Interrupt on Acknowledge Error 8 enabled.
- Bit 7 **AE7IE**: Acknowledge Error 7 Interrupt Enable  
This bit enables the interrupt generation on Acknowledge Error 7.  
0: Interrupt on Acknowledge Error 7 disabled.  
1: Interrupt on Acknowledge Error 7 enabled.
- Bit 6 **AE6IE**: Acknowledge Error 6 Interrupt Enable  
This bit enables the interrupt generation on Acknowledge Error 6.  
0: Interrupt on Acknowledge Error 6 disabled.  
1: Interrupt on Acknowledge Error 6 enabled.
- Bit 5 **AE5IE**: Acknowledge Error 5 Interrupt Enable  
This bit enables the interrupt generation on Acknowledge Error 5.  
0: Interrupt on Acknowledge Error 5 disabled.  
1: Interrupt on Acknowledge Error 5 enabled.
- Bit 4 **AE4IE**: Acknowledge Error 4 Interrupt Enable  
This bit enables the interrupt generation on Acknowledge Error 4.  
0: Interrupt on Acknowledge Error 4 disabled.  
1: Interrupt on Acknowledge Error 4 enabled.
- Bit 3 **AE3IE**: Acknowledge Error 3 Interrupt Enable  
This bit enables the interrupt generation on Acknowledge Error 3.  
0: Interrupt on Acknowledge Error 3 disabled.  
1: Interrupt on Acknowledge Error 3 enabled.
- Bit 2 **AE2IE**: Acknowledge Error 2 Interrupt Enable  
This bit enables the interrupt generation on Acknowledge Error 2.  
0: Interrupt on Acknowledge Error 2 disabled.  
1: Interrupt on Acknowledge Error 2 enabled.
- Bit 1 **AE1IE**: Acknowledge Error 1 Interrupt Enable  
This bit enables the interrupt generation on Acknowledge Error 1.  
0: Interrupt on Acknowledge Error 1 disabled.  
1: Interrupt on Acknowledge Error 1 enabled.
- Bit 0 **AE0IE**: Acknowledge Error 0 Interrupt Enable  
This bit enables the interrupt generation on Acknowledge Error 0.  
0: Interrupt on Acknowledge Error 0 disabled.  
1: Interrupt on Acknowledge Error 0 enabled.

### 29.15.44 DSI Host Interrupt Enable Register 1 (DSI\_IER1)

Address offset: 0x00C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	GPRXE IE	GPRDE IE	GPTXE IE	GPWRE IE	GCWR EIE	LPWR EIE	EOTPE IE	PSE IE	CRCE IE	ECCM EIE	ECCSE IE	TOLPRX IE	TOHSTX IE
			rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

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

- Bit 12 **GPRXEIE**: Generic Payload Receive Error Interrupt Enable  
 This bit enables the interrupt generation on Generic Payload Receive Error.  
 0: Interrupt on Generic Payload Receive Error disabled.  
 1: Interrupt on Generic Payload Receive Error enabled.
- Bit 11 **GPRDEIE**: Generic Payload Read Error Interrupt Enable  
 This bit enables the interrupt generation on Generic Payload Read Error.  
 0: Interrupt on Generic Payload Read Error disabled.  
 1: Interrupt on Generic Payload Read Error enabled.
- Bit 10 **GPTXEIE**: Generic Payload Transmit Error Interrupt Enable  
 This bit enables the interrupt generation on Generic Payload Transmit Error.  
 0: Interrupt on Generic Payload Transmit Error disabled.  
 1: Interrupt on Generic Payload Transmit Error enabled.
- Bit 9 **GPWREIE**: Generic Payload Write Error Interrupt Enable  
 This bit enables the interrupt generation on Generic Payload Write Error.  
 0: Interrupt on Generic Payload Write Error disabled.  
 1: Interrupt on Generic Payload Write Error enabled.
- Bit 8 **GCWREIE**: Generic Command Write Error Interrupt Enable  
 This bit enables the interrupt generation on Generic Command Write Error.  
 0: Interrupt on Generic Command Write Error disabled.  
 1: Interrupt on Generic Command Write Error enabled.
- Bit 7 **LPWREIE**: LTDC Payload Write Error Interrupt Enable  
 This bit enables the interrupt generation on LTDC Payload Write Error.  
 0: Interrupt on LTDC Payload Write Error disabled.  
 1: Interrupt on LTDC Payload Write Error enabled.
- Bit 6 **EOTPEIE**: EoTp Error Interrupt Enable  
 This bit enables the interrupt generation on EoTp Error.  
 0: Interrupt on EoTp Error disabled.  
 1: Interrupt on EoTp Error enabled.

- Bit 5 **PSEIE**: Packet Size Error Interrupt Enable  
 This bit enables the interrupt generation on Packet Size Error.  
 0: Interrupt on Packet Size Error disabled.  
 1: Interrupt on Packet Size Error enabled.
- Bit 4 **CRCEIE**: CRC Error Interrupt Enable  
 This bit enables the interrupt generation on CRC Error.  
 0: Interrupt on CRC Error disabled.  
 1: Interrupt on CRC Error enabled.
- Bit 3 **ECCMEIE**: ECC Multi-bit Error Interrupt Enable  
 This bit enables the interrupt generation on ECC Multi-bit Error.  
 0: Interrupt on ECC Multi-bit Error disabled.  
 1: Interrupt on ECC Multi-bit Error enabled.
- Bit 2 **ECCSEIE**: ECC Single-bit Error Interrupt Enable  
 This bit enables the interrupt generation on ECC Single-bit Error.  
 0: Interrupt on ECC Single-bit Error disabled.  
 1: Interrupt on ECC Single-bit Error enabled.
- Bit 1 **TOLPRXIE**: Timeout Low-Power Reception Interrupt Enable  
 This bit enables the interrupt generation on Timeout Low-Power Reception.  
 0: Interrupt on Timeout Low-Power Reception disabled.  
 1: Interrupt on Timeout Low-Power Reception enabled.
- Bit 0 **TOHSTXIE**: Timeout High-Speed Transmission Interrupt Enable  
 This bit enables the interrupt generation on Timeout High-Speed Transmission .  
 0: Interrupt on Timeout High-Speed Transmission disabled.  
 1: Interrupt on Timeout High-Speed Transmission enabled.

### 29.15.45 DSI Host Force Interrupt Register 0 (DSI\_FIR0)

Address offset: 0x00D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPE4	FPE3	FPE2	FPE1	FPE0
											wo	wo	wo	wo	wo
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAE15	FAE14	FAE13	FAE12	FAE11	FAE10	FAE9	FAE8	FAE7	FAE6	FAE5	FAE4	FAE3	FAE2	FAE1	FAE0
wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo

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

Bit 20 **FPE4**: Force PHY Error 4  
 Writing one to this bit forces a PHY Error 4.

Bit 19 **FPE3**: Force PHY Error 3  
 Writing one to this bit forces a PHY Error 3.

- Bit 18 **FPE2**: Force PHY Error 2  
Writing one to this bit forces a PHY Error 2.
- Bit 17 **FPE1**: Force PHY Error 1  
Writing one to this bit forces a PHY Error 1.
- Bit 16 **FPE0**: Force PHY Error 0  
Writing one to this bit forces a PHY Error 0.
- Bit 15 **FAE15**: Force Acknowledge Error 15  
Writing one to this bit forces an Acknowledge Error 15.
- Bit 14 **FAE14**: Force Acknowledge Error 14  
Writing one to this bit forces an Acknowledge Error 14.
- Bit 13 **FAE13**: Force Acknowledge Error 13  
Writing one to this bit forces an Acknowledge Error 13.
- Bit 12 **FAE12**: Force Acknowledge Error 12  
Writing one to this bit forces an Acknowledge Error 12.
- Bit 11 **FAE11**: Force Acknowledge Error 11  
Writing one to this bit forces an Acknowledge Error 11.
- Bit 10 **FAE10**: Force Acknowledge Error 10  
Writing one to this bit forces an Acknowledge Error 10.
- Bit 9 **FAE9**: Force Acknowledge Error 9  
Writing one to this bit forces an Acknowledge Error 9.
- Bit 8 **FAE8**: Force Acknowledge Error 8  
Writing one to this bit forces an Acknowledge Error 8.
- Bit 7 **FAE7**: Force Acknowledge Error 7  
Writing one to this bit forces an Acknowledge Error 7.
- Bit 6 **FAE6**: Force Acknowledge Error 6  
Writing one to this bit forces an Acknowledge Error 6.
- Bit 5 **FAE5**: Force Acknowledge Error 5  
Writing one to this bit forces an Acknowledge Error 5.
- Bit 4 **FAE4**: Force Acknowledge Error 4  
Writing one to this bit forces an Acknowledge Error 4.
- Bit 3 **FAE3**: Force Acknowledge Error 3  
Writing one to this bit forces an Acknowledge Error 3.
- Bit 2 **FAE2**: Force Acknowledge Error 2  
Writing one to this bit forces an Acknowledge Error 2.
- Bit 1 **FAE1**: Force Acknowledge Error 1  
Writing one to this bit forces an Acknowledge Error 1.
- Bit 0 **FAE0**: Force Acknowledge Error 0  
Writing one to this bit forces an Acknowledge Error 0.

**29.15.46 DSI Host Force Interrupt Register 1 (DSI\_FIR1)**

Address offset: 0x00DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FGP RXE	FGP RDE	FGP TXE	FGP WRE	FGC WRE	FLP WRE	FE OTPE	FPSE	FCRCE	FECC ME	FECC SE	FTOLP RX	FTOHS TX
			wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo

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

- Bit 12 **FGPRXE**: Force Generic Payload Receive Error  
Writing one to this bit forces a Generic Payload Receive Error.
- Bit 11 **FGPRDE**: Force Generic Payload Read Error  
Writing one to this bit forces a Generic Payload Read Error.
- Bit 10 **FGPTXE**: Force Generic Payload Transmit Error  
Writing one to this bit forces a Generic Payload Transmit Error.
- Bit 9 **FGPWRE**: Force Generic Payload Write Error  
Writing one to this bit forces a Generic Payload Write Error.
- Bit 8 **FGCWRE**: Force Generic Command Write Error  
Writing one to this bit forces a Generic Command Write Error.
- Bit 7 **FLPWRE**: Force LTDC Payload Write Error  
Writing one to this bit forces a LTDC Payload Write Error.
- Bit 6 **FEOTPE**: Force EoTp Error  
Writing one to this bit forces a EoTp Error.
- Bit 5 **FPSE**: Force Packet Size Error  
Writing one to this bit forces a Packet Size Error.
- Bit 4 **FCRCE**: Force CRC Error  
Writing one to this bit forces a CRC Error.
- Bit 3 **FECCME**: Force ECC Multi-bit Error  
Writing one to this bit forces a ECC Multi-bit Error.
- Bit 2 **FECCSE**: Force ECC Single-bit Error  
Writing one to this bit forces a ECC Single-bit Error.
- Bit 1 **FTOLPRX**: Force Timeout Low-Power Reception  
Writing one to this bit forces a Timeout Low-Power Reception.
- Bit 0 **FTOHSTX**: Force Timeout High-Speed Transmission  
Writing one to this bit forces a Timeout High-Speed Transmission.

### 29.15.47 DSI Host Video Shadow Control Register (DSI\_VSCR)

Address offset: 0x0100

Reset value: 0x0000 0000

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

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

Bit 8 **UR**: Update Register

When set to 1, the LTDC registers are copied to the auxiliary registers. After copying, this bit is auto cleared.

0: No update requested.

1: Register update requested.

Bits 7: 1 Reserved, must be kept at reset value

Bit 0 **EN**: Enable

When set to 1, DSI Host LTDC interface receives the active configuration from the auxiliary registers.

When this bit is set along with the UR bit, the auxiliary registers are automatically updated.

0: Register update is disabled.

1: Register update is enabled.

### 29.15.48 DSI Host LTDC Current VCID Register (DSI\_LCVCIDR)

Address offset: 0x010C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	VCID[1:0]	
														ro	

Bits 31: 2 Reserved, must be kept at reset value

Bits 1: 0 **VCID**: Virtual Channel ID

This field returns the Virtual Channel ID for the LTDC interface.

### 29.15.49 DSI Host LTDC Current Color Coding Register (DSI\_LCCCR)

Address offset: 0x0110

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	LPE	Res.	Res.	Res.	Res.	COLC[3:0]			
							ro								ro

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

Bit 8 **LPE**: Loosely Packed Enable

This bit returns the current state of the loosely packed variant to 18-bit configurations.

0: Loosely Packed variant disabled.

1: Loosely Packed variant enabled.

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

Bits 3: 0 **COLC**: Color Coding

This field returns the current LTDC interface color coding

0000: 16-bit configuration 1

0001: 16-bit configuration 2

0010: 16-bit configuration 3

0011: 18-bit configuration 1

0100: 18-bit configuration 2

0101: 24-bit

0110 - 1111: reserved

If LTDC interface in Command mode is chosen and currently works in the Command mode (CMDM=1), then 0110-1111: 24-bit

### 29.15.50 DSI Host Low-Power mode Current Configuration Register (DSI\_LPMCCR)

Address offset: 0x0118

Reset value: 0x0000 0000

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



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

Bits 23: 16 **LPSIZE**: Largest Packet Size

This field returns the current size, in bytes, of the largest packet that can fit in a line during VSA, VBP and VFP regions, for the transmission of commands in Low-Power mode.

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

Bits 7: 0 **VLPSIZE**: VACT Largest Packet Size

This field returns the current size, in bytes, of the largest packet that can fit in a line during VACT regions, for the transmission of commands in Low-Power mode.

### 29.15.51 DSI Host Video mode Current Configuration Register (DSI\_VMCCR)

Address offset: 0x0138

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	LPCE	FBTAAE	LPHFE	LPHBPE	LPVAE	LPVFPE	LPVBPE	LPVSAE	VMT[1:0]	
ro	ro	ro	ro	ro	ro	ro	ro								ro

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

Bit 9 **LPCE**: Low-Power Command Enable

This bit returns the current command transmission state in Low-Power mode.  
 0: Command transmission in Low-Power mode is disabled.  
 1: Command transmission in Low-Power mode is enabled.

Bit 8 **FBTAAE**: Frame BTA Acknowledge Enable

This bit returns the current state of request for an acknowledge response at the end of a frame.  
 0: Acknowledge response at the end of a frame is disabled.  
 1: Acknowledge response at the end of a frame is enabled.

Bit 7 **LPHFE**: Low-Power Horizontal Front-Porch Enable

This bit returns the current state of return to Low-Power inside the Horizontal Front Porch (HFP) period when timing allows.  
 0: Return to Low-Power inside the HFP period is disabled.  
 1: Return to Low-Power inside the HFP period is enabled.

Bit 6 **LPHBPE**: Low-power Horizontal Back-Porch Enable

This bit returns the current state of return to Low-Power inside the Horizontal Back Porch (HBP) period when timing allows.  
 0: Return to Low-Power inside the HBP period is disabled.  
 1: Return to Low-Power inside the HBP period is enabled.

- Bit 5 **LPVAE**: Low-Power Vertical Active Enable  
 This bit returns the current state of return to Low-Power inside the Vertical Active (VACT) period when timing allows.  
 0: Return to Low-Power inside the VACT is disabled.  
 1: Return to Low-Power inside the VACT is enabled.
- Bit 4 **LPVFPE**: Low-power Vertical Front-Porch Enable  
 This bit returns the current state of return to Low-Power inside the Vertical Front Porch (VFP) period when timing allows.  
 0: Return to Low-Power inside the VFP is disabled.  
 1: Return to Low-Power inside the VFP is enabled.
- Bit 3 **LPVBPE**: Low-power Vertical Back-Porch Enable  
 This bit returns the current state of return to Low-Power inside the Vertical Back Porch (VBP) period when timing allows.  
 0: Return to Low-Power inside the VBP is disabled.  
 1: Return to Low-Power inside the VBP is enabled.
- Bit 2 **LPVSAE**: Low-Power Vertical Sync time Enable  
 This bit returns the current state of return to Low-Power inside the Vertical Sync Time (VSA) period when timing allows.  
 0: Return to Low-Power inside the VSA is disabled.  
 1: Return to Low-Power inside the VSA is enabled
- Bits 1: 0 **VMT**: Video mode Type  
 This field returns the current video mode transmission type:  
 00: Non-Burst with sync pulses.  
 01: Non-Burst with sync events.  
 1x: Burst mode

### 29.15.52 DSI Host Video Packet Current Configuration Register (DSI\_VPCCR)

Address offset: 0x013C

Reset value: 0x0000 0000

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

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

Bits 13: 0 **VPSIZE**: Video Packet Size

This field returns the number of pixels in a single video packet.

**29.15.53 DSI Host Video Chunks Current Configuration Register (DSI\_VCCCR)**

Address offset: 0x0140

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	NUMC[12:0]												0
			ro												

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

Bits 12: 0 **NUMC**: Number of Chunks

This field returns the number of chunks being transmitted during a Line period.

**29.15.54 DSI Host Video Null Packet Current Configuration Register (DSI\_VNPCCR)**

Address offset: 0x0144

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	NPSIZE[12:0]												0
			ro												

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

Bits 12: 0 **NPSIZE**: Null Packet Size

This field returns the number of bytes inside a null packet.

**29.15.55 DSI Host Video HSA Current Configuration Register (DSI\_VHSACCR)**

Address offset: 0x0148

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	HSA[11:0]											
				ro											

Bits 31: 12 Reserved, must be kept at reset value

Bits 11: 0 **HSA**: Horizontal Synchronism Active duration

This fields returns the Horizontal Synchronism Active period in lane byte clock cycles.

**29.15.56 DSI Host Video HBP Current Configuration Register (DSI\_VHBPCCR)**

Address offset: 0x014C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	HBP[11:0]											
				ro											

Bits 31: 12 Reserved, must be kept at reset value

Bits 11: 0 **HBP**: Horizontal Back-Porch duration

This fields returns the Horizontal Back-Porch period in lane byte clock cycles.

**29.15.57 DSI Host Video Line Current Configuration Register (DSI\_VLCCR)**

Address offset: 0x0150

Reset value: 0x0000 0000

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

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

Bits 14: 0 **HLINE**: Horizontal Line duration

This fields return the current total of the Horizontal Line period (HSA+HBP+HACT+HFP) counted in lane byte clock cycles.

**29.15.58 DSI Host Video VSA Current Configuration Register (DSI\_VVSACCR)**

Address offset: 0x0154

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	VSA[9:0]									
ro															

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

Bits 9: 0 **VSA**: Vertical Synchronism Active duration

This fields return the current Vertical Synchronism Active period measured in number of horizontal lines.

**29.15.59 DSI Host Video VBP Current Configuration Register (DSI\_VBPCCR)**

Address offset: 0x0158

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	VBP[9:0]									
						ro									

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

Bits 9: 0 **VBP**: Vertical Back-Porch duration

This fields returns the current Vertical Back-Porch period measured in number of horizontal lines.

**29.15.60 DSI Host Video VFP Current Configuration Register (DSI\_VVFPCCR)**

Address offset: 0x015C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	VFP[9:0]									
						ro									

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

Bits 9: 0 **VFP**: Vertical Front-Porch duration

This fields returns the current Vertical Front-Porch period measured in number of horizontal lines.

**29.15.61 DSI Host Video VA Current Configuration Register (DSI\_VVACCR)**

Address offset: 0x0160

Reset value: 0x0000 0000

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

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

Bits 13: 0 **VA**: Vertical Active duration

This fields returns the current Vertical Active period measured in number of horizontal lines.

## 29.16 DSI Wrapper Registers

### 29.16.1 DSI Wrapper Configuration Register (DSI\_WCFGR)

Address offset: 0x0400

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	VSPOL	AR	TEPOL	TESRC	COLMUX[2:0]		DSIM	
								rw	rw	rw	rw	rw		rw	

Bits 31: 8 Reserved

Bit 7 **VSPOL**: VSync Polarity

Select the VSync edge on which the LTDC is halted

0: LTDC halted on a falling edge.

1: LTDC halted on a rising edge.

This bit shall only be changed when DSI is stopped (CR.DSIEN = 0 and CR.EN = 0).

Bit 6 **AR**: Automatic Refresh

Selects the refresh mode in DBI mode

0: automatic refresh mode disabled.

1: automatic refresh mode enabled.

This bit shall only be changed when DSI Host is stopped (CR.EN = 0).

Bit 5 **TEPOL**: TE Polarity

Selects the polarity of the external pin Tearing Effect (TE) source

0: rising edge.

1: falling edge.

This bit shall only be changed when DSI Host is stopped (CR.EN = 0).

Bit 4 **TESRC**: TE Source

Selects the Tearing Effect (TE) source

0: DSI Link.

1: External pin.

This bit shall only be changed when DSI Host is stopped (CR.EN = 0).



- Bits 3: 1 **COLMUX**: Color Multiplexing  
 Selects the color multiplexing used by DSI Host  
 000: 16-bit configuration 1  
 001: 16-bit configuration 2  
 010: 16-bit configuration 3  
 011: 18-bit configuration 1  
 100: 18-bit configuration 2  
 101: 24-bit  
 This field shall only be changed when DSI is stopped (CR.DSIEN = 0 and CR.EN = 0).
- Bit 0 **DSIM**: DSI mode  
 Selects the mode for the video transmission  
 0: Video mode.  
 1: Adapted Command mode.  
 This bit shall only be changed when DSI Host is stopped (CR.EN = 0).

### 29.16.2 DSI Wrapper Control Register (DSI\_WCR)

Address offset: 0x0404

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	DSIEN	LTDCEN	SHTDN	COLM
												rw	w1o	rw	rw

- Bits 31: 4 Reserved
- Bit 3 **DSIEN**: DSI Enable  
 Enables the DSI wrapper  
 0: DSI is disabled.  
 1: DSI is enabled.
- Bit 2 **LTDCEN**: LTDC Enable  
 Enables the LTDC for a frame transfer in Adapted Command mode  
 0: LTDC is disabled.  
 1: LTDC is enabled.
- Bit 1 **SHTDN**: Shutdown  
 Controls the display shutdown in Video mode:  
 0: display ON.  
 1: display OFF.
- Bit 0 **COLM**: Color mode  
 Controls the display color mode in Video mode:  
 0: Full color mode.  
 1: Eight color mode.

### 29.16.3 DSI Wrapper Interrupt Enable Register (DSI\_WIER)

Address offset: 0x0408

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RRIE	Res.	Res.	PLLUIE	PLLIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERIE	TEIE
		rw			rw	rw								rw	rw

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

Bit 13 **RRIE**: Regulator Ready Interrupt Enable  
 Enables the Regulator Ready interrupt  
 0: Regulator Ready interrupt disabled.  
 1: Regulator Ready interrupt enabled.

Bits 12:11 Reserved, must be kept at reset value

Bit 10 **PLLUIE**: PLL Unlock Interrupt Enable  
 Enables the PLL Unlock interrupt  
 0: PLL Unlock interrupt disabled.  
 1: PLL Unlock interrupt enabled.

Bit 9 **PLLIE**: PLL Lock Interrupt Enable  
 Enables the PLL Lock interrupt  
 0: PLL Lock interrupt disabled.  
 1: PLL Lock interrupt enabled.

Bits 8:2 Reserved, must be kept at reset value

Bit 1 **ERIE**: End of Refresh Interrupt Enable  
 Enables the End of Refresh interrupt  
 0: End of refresh interrupt disabled.  
 1: End of refresh interrupt enabled.

Bit 0 **TEIE**: Tearing Effect Interrupt Enable  
 Enables the Tearing Effect interrupt  
 0: Tearing Effect interrupt disabled.  
 1: Tearing Effect interrupt enabled.

### 29.16.4 DSI Wrapper Interrupt & Status Register (DSI\_WISR)

Address offset: 0x040C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RRIF	RRS	Res.	PLLUIF	PLLIF	PLLS	Res.	Res.	Res.	Res.	Res.	BUSY	ERIF	TEIF
		r	r		r	r	r						r	r	r

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

Bit 13 **RRIF**: Regulator Ready Interrupt Flag

This bit is set when the regulator becomes ready:

0: No regulator ready event occurred.

1: Regulator ready event occurred.

Bit 12 **RRS**: Regulator Ready Status

This bit gives the status of the regulator:

0: Regulator is not ready.

1: Regulator is ready.

Bit 11 Reserved, must be kept at reset value

Bit 10 **PLLUIF**: PLL Unlock Interrupt Flag

This bit is set when the PLL becomes unlocked:

0: No PLL unlock event occurred.

1: PLL unlock event occurred.

Bit 9 **PLLIF**: PLL Lock Interrupt Flag

This bit is set when the PLL becomes locked:

0: No PLL lock event occurred.

1: PLL lock event occurred.

Bit 8 **PLLS**: PLL Lock Status

This bit is set when the PLL is locked and cleared when it is unlocked:

0: PLL is unlocked.

1: PLL is locked.

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

Bit 2 **BUSY**: Busy Flag

This bit is set when the transfer of a frame in Adapted Command mode is ongoing:

0: No transfer on going

1: Transfer on going

- Bit 1 **ERIF**: End of Refresh Interrupt Flag  
 This bit is set when the transfer of a frame in Adapted Command mode is finished:  
 0: No end of refresh event occurred.  
 1: End of refresh event occurred.
- Bit 0 **TEIF**: Tearing Effect Interrupt Flag  
 This bit is set when a tearing effect event occurs  
 0: No tearing effect event occurred.  
 1: Tearing effect event occurred.

### 29.16.5 DSI Wrapper Interrupt Flag Clear Register (DSI\_WIFCR)

Address offset: 0x0410

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CRRIF	Res.	Res.	CPLLUIF	CPLLIF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERIF	CTEIF
		w1o			w1o	w1o								w1o	w1o

- Bits 31: 14 Reserved, must be kept at reset value
- Bit 13 **CRRIF**: Clear Regulator Ready Interrupt Flag  
 Write 1 clears the RRIF flag in the DSI\_WSR register
- Bits 12: 11 Reserved, must be kept at reset value
- Bit 10 **CPLLUIF**: Clear PLL Unlock Interrupt Flag  
 Write 1 clears the PLLUIF flag in the DSI\_WSR register
- Bit 9 **CPLLIF**: Clear PLL Lock Interrupt Flag  
 Write 1 clears the PLLIF flag in the DSI\_WSR register
- Bits 8: 2 Reserved, must be kept at reset value
- Bit 1 **CERIF**: Clear End of Refresh Interrupt Flag  
 Write 1 clears the ERIF flag in the DSI\_WSR register
- Bit 0 **CTEIF**: Clear Tearing Effect Interrupt Flag  
 Write 1 clears the TEIF flag in the DSI\_WSR register

### 29.16.6 DSI Wrapper PHY Configuration Register 0 (DSI\_WPCR0)

Address offset: 0x0418

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TCLK POSTEN	TLPXC EN	THSEXIT EN	TLPXD EN	THSZEROEN	THSTRAILEN	THSPREPEN	TCLKZEROEN	TCLKPREPEN	PDEN	Res.	TDDL
				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w		r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CDOFF DL	FTXS MDL	FTXS MCL	HSIDL1	HSIDL0	HSICL	SWDL1	SWDL0	SWCL	UIX4[5:0]					
	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w					

Bits 31: 28 Reserved

Bit 27 **TCLKPOSTEN**: custom time for  $t_{CLK-POST}$  Enable

This bit enable the manual programming of  $t_{CLK-POST}$  duration in the D-PHY. The desired value must be programmed in the TCLKPOST field of the DSI\_WPCR4 register.

0: Default value is used for  $t_{CLKPOST}$

1: Programmable value is used for  $t_{CLKPOST}$

Bit 26 **TLPXCEN**: custom time for  $t_{LPX}$  for Clock lane Enable

This bit enable the manual programming of  $t_{LPX}$  duration for the clock lane in the D-PHY. The desired value must be programmed in the TLPXC field of the DSI\_WPCR3 register.

0: Default value is used for  $t_{LPX}$  for the clock lane.

1: Programmable value is used for  $t_{LPX}$  for the clock lane.

Bit 25 **THSEXITEN**: custom time for  $t_{HS-EXIT}$  Enable

This bit enable the manual programming of  $t_{HS-EXIT}$  duration in the D-PHY. The desired value must be programmed in the THSEXIT field of the DSI\_WPCR3 register.

0: Default value is used for  $t_{HS-EXIT}$

1: Programmable value is used for  $t_{HS-EXIT}$

Bit 24 **TLPXDEN**: custom time for  $t_{LPX}$  for Data lanes Enable

This bit enable the manual programming of  $T_{LPX}$  duration for the data lanes in the D-PHY. The desired value must be programmed in the TLPXD field of the DSI\_WPCR3 register.

0: Default value is used for  $T_{LPX}$  for the data lanes.

1: Programmable value is used for  $T_{LPX}$  for the data lanes.

Bit 23 **THSZEROEN**: custom time for  $t_{HS-ZERO}$  Enable

This bit enable the manual programming of  $t_{HS-ZERO}$  duration in the D-PHY. The desired value must be programmed in the THSZERO field of the DSI\_WPCR3 register.

0: Default value is used for  $t_{HS-ZERO}$

1: Programmable value is used for  $t_{HS-ZERO}$

Bit 22 **THSTRAILEN**: custom time for  $t_{HS-TRAIL}$  Enable

This bit enable the manual programming of  $T_{HS-TRAIL}$  duration in the D-PHY. The desired value must be programmed in the THSTRAIL field of the DSI\_WPCR2 register.

0: Default value is used for  $T_{HS-TRAIL}$

1: Programmable value is used for  $T_{HS-TRAIL}$

- Bit 21 **THSPREPEN**: custom time for  $t_{HS-PREPARE}$  Enable  
 This bit enable the manual programming of  $t_{HS-PREPARE}$  duration in the D-PHY. The desired value must be programmed in the THSPREP field of the DSI\_WPCR2 register.  
 0: Default value is used for  $t_{HS-PREPARE}$   
 1: Programmable value is used for  $t_{HS-PREPARE}$
- Bit 20 **TCLKZEROEN**: custom time for  $t_{CLK-ZERO}$  Enable  
 This bit enable the manual programming of  $t_{CLK-ZERO}$  duration in the D-PHY. The desired value must be programmed in the TCLKZERO field of the DSI\_WPCR2 register.  
 0: Default value is used for  $t_{CLK-ZERO}$   
 1: Programmable value is used for  $t_{CLK-ZERO}$
- Bit 19 **TCLKPREPEN**: custom time for  $t_{CLK-PREPARE}$  Enable  
 This bit enable the manual programming of  $t_{CLK-PREPARE}$  duration in the D-PHY. The desired value must be programmed in the TLKCPREP field of the DSI\_WPCR2 register.  
 0: Default value is used for  $t_{CLK-PREPARE}$   
 1: Programmable value is used for  $t_{CLK-PREPARE}$
- Bit 18 **PDEN**: Pull-Down Enable  
 This bit enables a pull-down on the lane to prevent from floating states when unused:  
 0: Pull-down on lanes disabled  
 1: Pull-down on lanes enabled
- Bit 17 Reserved
- Bit 16 **TDDL**: Turn Disable Data Lanes  
 This bit forces the Data Lane to remain in RX event if it receives a bus turn around request from the other side:  
 0: No effect.  
 1: Force Data Lanes in RX mode after a BTA.
- Bit 15 Reserved
- Bit 14 **CDOFFDL**: Contention Detection OFF on Data Lanes  
 When only forward Escape mode is used, this signal can be made high to switch off the contention detector and reduce static power consumption:  
 0: Contention Detection on Data Lane ON.  
 1: Contention Detection on Data Lane OFF.
- Bit 13 **FTXSMDL**: Force in TX Stop mode the Data Lanes  
 This bit forces the Data Lanes in TX stop mode. It is used to initialize a lane module in Transmit mode. It causes the lane module to immediately jump to transmit Control mode and to begin transmitting a stop state (LP-11). It can be used to go back in TX mode after a wrong BTA sequence:  
 0: No effect.  
 1: Force the Data Lanes in TX Stop mode.
- Bit 12 **FTXSMCL**: Force in TX Stop mode the Clock Lane  
 This bit forces the Clock Lane in TX stop mode. It is used to initialize a lane module in Transmit mode. It causes the lane module to immediately jump to transmit Control mode and to begin transmitting a stop state (LP-11). It can be used to go back in TX mode after a wrong BTA sequence:  
 0: No effect.  
 1: Force the Clock Lane in TX Stop mode.

- Bit 11 **HSIDL1**: Invert the High-Speed data signal on Data Lane 1  
 This bit invert the High-Speed data signal on data lane 1:  
 0: Normal data signal configuration.  
 1: Inverted data signal configuration.
- Bit 10 **HSIDL0**: Invert the High-Speed data signal on Data Lane 0  
 This bit invert the High-Speed data signal on clock lane:  
 0: Normal data signal configuration.  
 1: Inverted data signal configuration.
- Bit 9 **HSICL**: Invert High-Speed data signal on Clock Lane  
 This bit invert the High-Speed data signal on clock lane:  
 0: Normal data configuration.  
 1: Inverted data configuration.
- Bit 8 **SWDL1**: Swap Data Lane 1 pins  
 This bit swap the pins on clock lane  
 0: Regular clock lane pin configuration.  
 1: Swapped clock lane pin.
- Bit 7 **SWDL0**: Swap Data Lane 0 pins  
 This bit swap the pins on data lane 0:  
 0: Regular clock lane pin configuration.  
 1: Swapped clock lane pin.
- Bit 6 **SWCL**: Swap Clock Lane pins  
 This bit swap the pins on clock lane:  
 0: Regular clock lane pin configuration.  
 1: Swapped clock lane pin.
- Bits 5:0 **UIX4**: Unit Interval multiplied by 4  
 This field defines the bit period in High-Speed mode in unit of 0.25 ns.  
 As an example, if the unit interval is 3ns, a value of twelve (0x0C) should be driven to this input. This value is used to generate delays. If the period is not a multiple of 0.25ns, the value driven should be rounded down. For example, a 600Mbit/s link uses a unit interval of 1.667 ns. Multiplying by four results in 6.667. In this case, a value of 6 (not 7) should be driven onto the ui\_x4 input.

### 29.16.7 DSI Wrapper PHY Configuration Register 1 (DSI\_WPCR1)

Address offset: 0x041C

Reset value: 0x0000 0000

*Note:* This register shall be programmed only when DSI is stopped (CR.DSIEN=0 and CR.EN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	LPRXFT[1:0]	Res.	Res.	FLPRXLPM	Res.	Res.	HSTXSRCDL[1:0]	HSTXSRCL[1:0]			
					rw			rw			rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	SDDC	Res.	Res.	LPSRDL[1:0]	LPSRCL[1:0]	Res.	Res.	HSTXDDL[1:0]	HSTXDCL[1:0]				
			rw			rw	rw			rw	rw				



Bits 31: 27 Reserved

Bits 26:25 **LPRXFT**: Low-Power RX low-pass Filtering Tuning

This signal can be used to tune the cutoff frequency of low-pass filter at the input of LPRX.

Bits 24: 23 Reserved

Bit 22 **FLPRXLPM**: Forces LP Receiver in Low-Power mode

This bit enables the Low-Power mode of LP receiver (LPRX). When set, the LPRX operates in Low-Power mode all the time (when this is not activated, LPRX operates in Low-Power mode during ULPS only):

0: No effect.

1: LPRX is forced in Low-Power mode.

Bits 21: 20 Reserved

Bits 19:18 **HSTXSRCDL**: High-Speed Transmission Slew Rate Control on Data Lanes

Slewrates Control for High Speed Transmitter Output. It can be used to change slew rate of Data lane HS transitions.

Default value should be '00'.

Bits 17:16 **HSTXSRCCL**: High-Speed Transmission Slew Rate Control on Clock Lane

Slewrates Control for High Speed Transmitter Output. It can be used to change slew rate of Clock lane HS transitions.

Default value should be '00'.

Bits 15:13 Reserved

Bit 12 **SDDC**: SDD Control

Switch on the additional current path to meet the SDDTx parameter defined by MIPI® D-PHY Specification on both clock and data lanes.

0: No effect.

1: Activate additional current path on all lanes.

Bits 11: 10 Reserved

Bits 9:8 **LPSRCDL**: Low-Power transmission Slew Rate Compensation on Data Lanes

Can be used to change slew rate of Data lane LP transitions.

Default value should be '00'.

Bits 7:6 **LPSRCCL**: Low-Power transmission Slew Rate Compensation on Clock Lane

Can be used to change slew rate of Clock lane LP transitions.

Default value should be '00'.

Bits 5: 4 Reserved

Bits 3:2 **HSTXDDL**: High-Speed Transmission Delay on Data Lanes

Delay tuner control to change delay (up to DP/DN) in data path. Can be used to change data edge transition positions with respect to clock edge on DP/DN.

Default value should be '00'.

Bits 1:0 **HSTXDCL**: High-Speed Transmission Delay on Clock Lane

Delay tuner control to change delay (upto DP/DN) in clock path. Can be used to change clock edge position with respect to data bit transitions on DP/DN.

Default value should be '00'.



### 29.16.8 DSI Wrapper PHY Configuration Register 2 (DSI\_WPCR2)

Address offset: 0x0420

Reset value: 0x0000 0000

*Note:* This register shall be programmed only when DSI is stopped (CR.DSIEN=0 and CR.EN = 0).

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

- Bits 31:24 **THSTRAIL**:  $t_{HS-TRAIL}$   
 This field defines the  $t_{HS-TRAIL}$  has specified in the MIPI® D-PHY specification. This value is used by the D-PHY when the THSTRAILEN bit of the DSI\_WPCR0 is set.  
 $THSTRAIL = 2 \times t_{HS-TRAIL}$  expressed in ns. The default value used by the D-PHY when THSTRAILEN bit of the DSI\_WPCR0 is reset is 140, i.e.  $70 \text{ ns} + 8 \times UI$ .
- Bits 23:16 **THSPREP**:  $t_{HS-PREPARE}$   
 This field defines the  $t_{HS-PREPARE}$  has specified in the MIPI® D-PHY specification. This value is used by the D-PHY when the THSPREPEN bit of the DSI\_WPCR0 is set.  
 $THSPREP = 2 \times t_{HS-PREPARE}$  expressed in ns. The default value used by the D-PHY when THSPREPEN bit of the DSI\_WPCR0 is reset is 126, i.e.  $63 \text{ ns} + 12 \times UI$ .
- Bits 15:8 **TCLKZERO**:  $t_{CLK-ZERO}$   
 This field defines the  $t_{CLK-ZERO}$  has specified in the MIPI® D-PHY specification. This value is used by the D-PHY when the TCLKZEROEN bit of the DSI\_WPCR0 is set.  
 $TCLKZERO = t_{CLK-ZERO} / 2$  expressed in ns. The default value used by the D-PHY when TCLKZEROEN bit of the DSI\_WPCR0 is reset is 195, i.e.  $390 \text{ ns}$ .
- Bits 7:0 **TCLKPREP**:  $t_{CLK-PREPARE}$   
 This field defines the  $t_{CLK-PREPARE}$  has specified in the MIPI® D-PHY specification. This value is used by the D-PHY when the TCLKPREPEN bit of the DSI\_WPCR0 is set.  
 $TCLKPREP = 2 \times t_{CLK-PREPARE}$  expressed in ns. The default value used by the D-PHY when TCLKPREPEN bit of the DSI\_WPCR0 is reset is 120, i.e.  $60 \text{ ns} + 20 \times UI$ .

### 29.16.9 DSI Wrapper PHY Configuration Register 3 (DSI\_WPCR4)

Address offs Vet: 0x0424

Reset value: 0x0000 0000

*Note:* This register shall be programmed only when DSI is stopped (CR. DSIEN=0 and DSI\_CR.EN = 0).

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

Bits 31:24 **TLPXC**:  $t_{LPXC}$  for Clock lane

This field defines the  $t_{LPX}$  has specified in the MIPI® D-PHY specification for the clock lane. This value is used by the D-PHY when the TLPXCEN bit of the DSI\_WPCR1 is set.  $TLPXC = 2 \times t_{LPX}$  expressed in ns. The default value used by the D-PHY when TLPXCEN bit of the DSI\_WPCR1 is reset is 100, i.e. 50 ns.

Bits 23:16 **THSEXIT**:  $t_{HSEXIT}$

This field defines the  $t_{HS-EXIT}$  has specified in the MIPI® D-PHY specification. This value is used by the D-PHY when the THSEXITEN bit of the DSI\_WPCR1 is set.  $THSEXIT = t_{HS-ZERO}$  expressed in ns. The default value used by the D-PHY when THSEXITEN bit of the DSI\_WPCR1 is reset is 100, i.e. 100 ns.

Bits 15:8 **TLPXD**:  $t_{LPX}$  for Data lanes

This field defines the  $t_{LPX}$  has specified in the MIPI® D-PHY specification for the data lanes. This value is used by the D-PHY when the TLPXDEN bit of the DSI\_WPCR1 is set.  $TLPXD = 2 \times t_{LPX}$  expressed in ns. The default value used by the D-PHY when TLPXDEN bit of the DSI\_WPCR1 is reset is 100, i.e. 50 ns.

Bits 7:0 **THSZERO**:  $t_{HS-ZERO}$

This field defines the  $t_{HS-ZERO}$  has specified in the MIPI® D-PHY specification. This value is used by the D-PHY when the THSZEROEN bit of the DSI\_WPCR1 is set.  $THSZERO = t_{HS-ZERO}$  expressed in ns. The default value used by the D-PHY when THSZEROEN bit of the DSI\_WPCR1 is reset is 175, i.e. 175 ns.

### 29.16.10 DSI Wrapper PHY Configuration Register 4 (DSI\_WPCR4)

Address offset: 0x0428

Reset value: 0x0000 0000

*Note:* This register shall be programmed only when DSI is stopped (CR.DSIEN=0 and DSI\_CR.EN = 0).

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

Bits 31: 8 Reserved

Bits 7:0 **TCLKPOST**:  $t_{CLK-POST}$

This field defines the  $t_{CLK-POST}$  has specified in the MIPI® D-PHY specification. This value is used by the D-PHY when the TCLKPOSTEN bit of the DSI\_WPCR0 is set.

$TCLKPOST = 2 \times t_{CLK-POST}$  expressed in ns. The default value used by the D-PHY when TCLKPOSTEN bit of the DSI\_WPCR0 is reset is 200, i.e.  $100 \text{ ns} + 120 * UI$ .

### 29.16.11 DSI Wrapper Regulator and PLL Control Register (DSI\_WRPCR)

Address offset: 0x0430

Reset value: 0x0000 0000

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

Bits 31: 25 Reserved

Bit 24 **REGEN**: Regulator Enable

This bit enables the DPHY regulator:

0: regulator disabled

1: regulator enabled

Bits 23: 18 Reserved

Bits 17:16 **ODF**: PLL Output Division Factor

This field configures the PLL Output Division Factor:

00: PLL output divided by 1.

01: PLL output divided by 2.

10: PLL output divided by 4.

11: PLL output divided by 8.



Bit 15 Reserved

Bits 14:11 **IDF**: PLL Input Division Factor

This field configures the PLL Input Division Factor:

000: PLL input divided by 1.

001: PLL input divided by 1.

010: PLL input divided by 2.

011: PLL input divided by 3.

100: PLL input divided by 4.

101: PLL input divided by 5.

110: PLL input divided by 6.

111: PLL input divided by 7.

Bits 10: 9 Reserved

Bits 8:2 **NDIV**: PLL Loop Division Factor

This field configures the PLL Loop Division Factor:

0 to 9: Reserved.

10 to 125: Allowed loop division factor values.

126 to 127: Reserved.

Bit 1 Reserved

Bit 0 **PLEN**: PLL Enable

This bit enables the D-PHY PLL:

0: PLL disable.

1: PLL enable.

## 29.17 DSI Host register map

The following table summarizes the DSI Host registers. Refer to the register boundary addresses table for the base addresses.

**Table 187. DSIHOST 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		
0x0000	<b>DSI_VR</b>	VERSION[31:0]																																	
	Reset value	0	0	1	1	0	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	1	0	1	1	0	0	
0x0004	<b>DSI_CR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN		
	Reset value																																	0	
0x0008	<b>DSIHSOT_CCR</b>	TOCKDIV[7:0]														TXECKDIV[7:0]																			
	Reset value																																		
0x000C	<b>DSI_LVCIDR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCID[1:0]		
	Reset value																																		0
0x0010	<b>DSI_LCOLCR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COLC[3:0]		
	Reset value																																		0
0x0014	<b>DSI_LPCR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSP	VSP	DEP	
	Reset value																																	0	0
0x0018	<b>DSI_LPMCR</b>	LPSIZE[7:0]														VLPSIZE[7:0]																			
	Reset value																																		
0x001C-0x0028		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
0x002C	<b>DSI_PCR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																		
0x0030	<b>DSI_GVCIDR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCID[1:0]	
	Reset value																																		
0x0034	<b>DSI_MCR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																		
0x0038	<b>DSI_VMCR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																		
0x003C	<b>DSI_VPCR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																		
0x0040	<b>DSI_VCCR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																		
0x0044	<b>DSI_VNPCR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																		
0x0048	<b>DSI_VHSACR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																		



Table 187. DSIHOST 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										
0x009C	<b>DSI_DLTCR</b> Reset value	HS2LP_TIME[7:0]								LP2HS_TIME[7:0]								MRD_TIME[14:0]								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A0	<b>DSI_PCTLR</b> 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.	Res.							
0x00A4	<b>DSI_PCONFR</b> 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.	Res.							
0x00A8	<b>DSI_PUCR</b> 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.	Res.							
0x00AC	<b>DSI_PTCR</b> 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.	Res.							
0x00B0	<b>DSI_PSR</b> 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.	Res.							
0x00B4 - 0x00B8		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.							
0x00BC	<b>DSI_ISR0</b> Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PE4	PE3	PE2	PE1	PE0	AE15	AE14	AE13	AE12	AE11	AE10	AE9	AE8	AE7	AE6	AE5	AE4	AE3	AE2	AE1	AE0	Res.									
0x00C0	<b>DSI_ISR1</b> 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.	Res.							
0x00C4	<b>DSI_IER0</b> Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PE4IE	PE3IE	PE2IE	PE1IE	PE0IE	AE15IE	AE14IE	AE13IE	AE12IE	AE11IE	AE10IE	AE9IE	AE8IE	AE7IE	AE6IE	AE5IE	AE4IE	AE3IE	AE2IE	AE1IE	AE0IE	Res.									
0x00C8	<b>DSI_IER1</b> 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.	Res.							
0x00CC-0x00D4		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.							
0x00D8	<b>DSI_FIR0</b> Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPE4	FPE3	FPE2	FPE1	FPE0	FAE15	FAE14	FAE13	FAE12	FAE11	FAE10	FAE9	FAE8	FAE7	FAE6	FAE5	FAE4	FAE3	FAE2	FAE1	FAE0	Res.									
0x00DC	<b>DSI_FIR1</b> 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.	Res.							
0x00E0 - 0x00FF		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.							
0x0100	<b>DSI_VSCR</b> 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.	Res.							







Table 187. DSIHOST 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			
0x0400	<b>DSI_WCFGR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value																										0	0	0	0	0	0	0	0	0	
0x0404	<b>DSI_WCR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value																															0	0	0	0	0
0x0408	<b>DSI_WIER</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																							0	0	0	0	0	0	0	0	0	0	0	0	0
0x040C	<b>DSI_WISR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																0	0	0	0
0x0410	<b>DSI_WIFCR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																			
0x0414		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
0x0418	<b>DSI_WPCR0</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x041C	<b>DSI_WPCR1</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value							0	0																											
0x0420	<b>DSI_WPCR2</b>	THSTRAIL[7:0]							THSPREP[7:0]							TCLKZEO[7:0]							TCLKPREP[7:0]													
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0424	<b>DSI_WPCR3</b>	TLPXC[7:0]							THSEXIT[7:0]							TLPXD[7:0]							THSZERO[7:0]													
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0428	<b>DSI_WPCR4</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																			
0x042C		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
0x0430	<b>DSI_WRPCR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value							0																												

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.



## 30 Touch sensing controller (TSC)

### 30.1 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.

### 30.2 TSC main features

The touch sensing controller has the following main features:

- Proven and robust surface charge transfer acquisition principle
- Supports up to 24 capacitive sensing channels
- Up to 8 capacitive sensing channels can be acquired in parallel offering a very good response time
- Spread spectrum feature to improve system robustness in noisy environments
- full hardware management of the charge transfer acquisition sequence
- Programmable charge transfer frequency
- Programmable sampling capacitor I/O pin
- Programmable channel I/O pin
- Programmable max count value to avoid long acquisition when a channel is faulty
- Dedicated end of acquisition and max count error flags with interrupt capability
- One sampling capacitor for up to 3 capacitive sensing channels to reduce the system components
- Compatible with proximity, touchkey, linear and rotary touch sensor implementation
- Designed to operate with STMTouch touch sensing firmware library

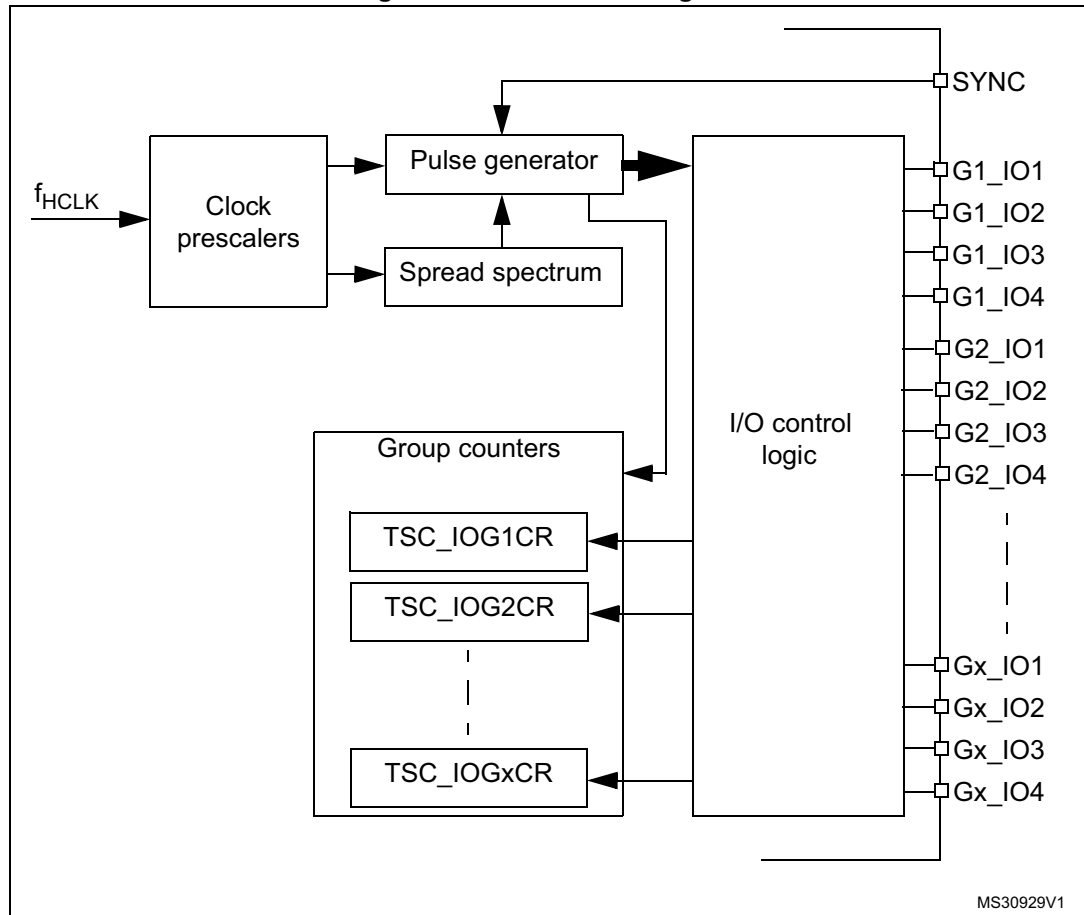
*Note: The number of capacitive sensing channels is dependent on the size of the packages and subject to IO availability.*

### 30.3 TSC functional description

#### 30.3.1 TSC block diagram

The block diagram of the touch sensing controller is shown in [Figure 213: TSC block diagram](#).

Figure 213. TSC block diagram



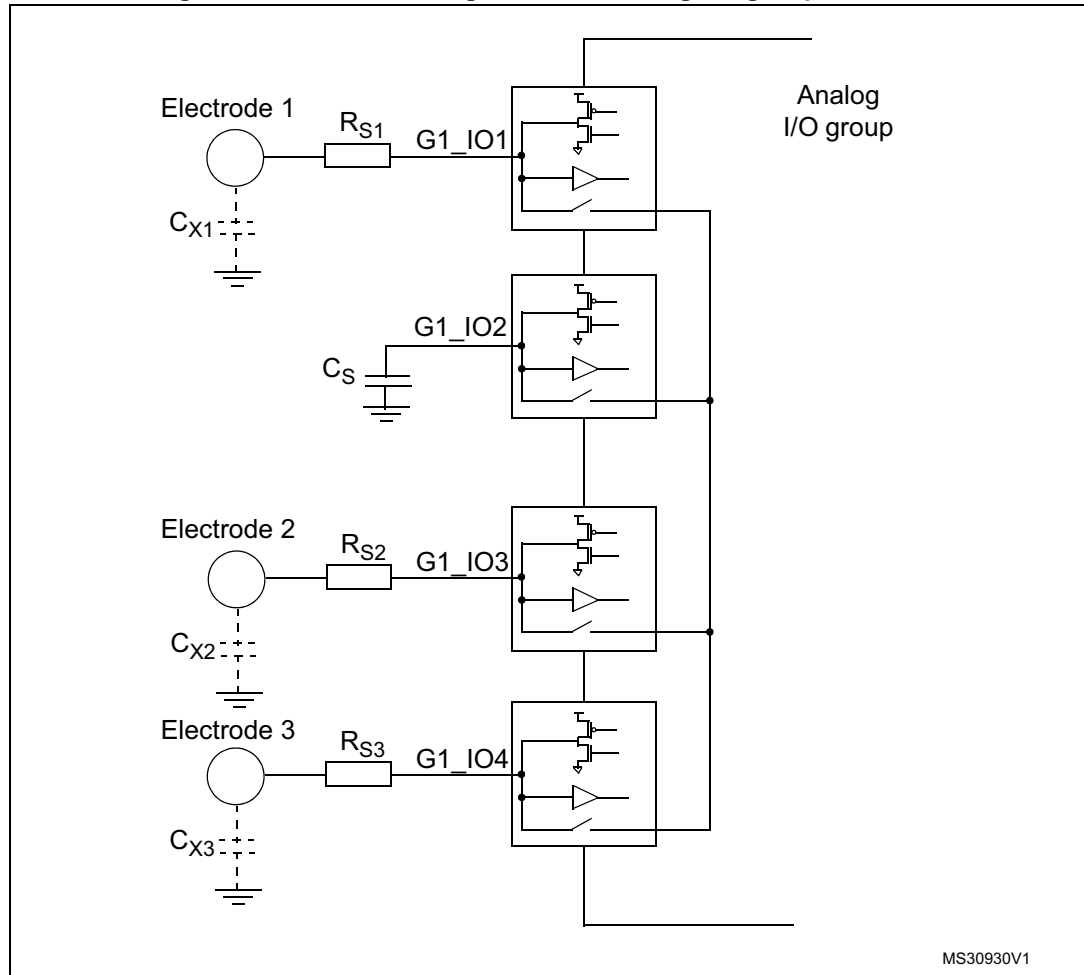
#### 30.3.2 Surface charge transfer acquisition overview

The surface charge transfer acquisition is a proven, robust and efficient way to measure a capacitance. It uses a minimum number of external components to operate with a single ended electrode type. This acquisition is designed around an analog I/O group which is composed of four GPIOs (see [Figure 214](#)). 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 214. Surface charge transfer analog I/O group structure**



*Note:*  $G_x\_IO_y$  where  $x$  is the analog I/O group number and  $y$  the GPIO number within the selected group.

The surface charge transfer acquisition principle consists of charging an electrode capacitance ( $C_X$ ) and transferring a part of the accumulated charge into a sampling capacitor ( $C_S$ ). This sequence is repeated until the voltage across  $C_S$  reaches a given threshold ( $V_{IH}$  in our case). The number of charge transfers required to reach the threshold is a direct representation of the size of the electrode capacitance.

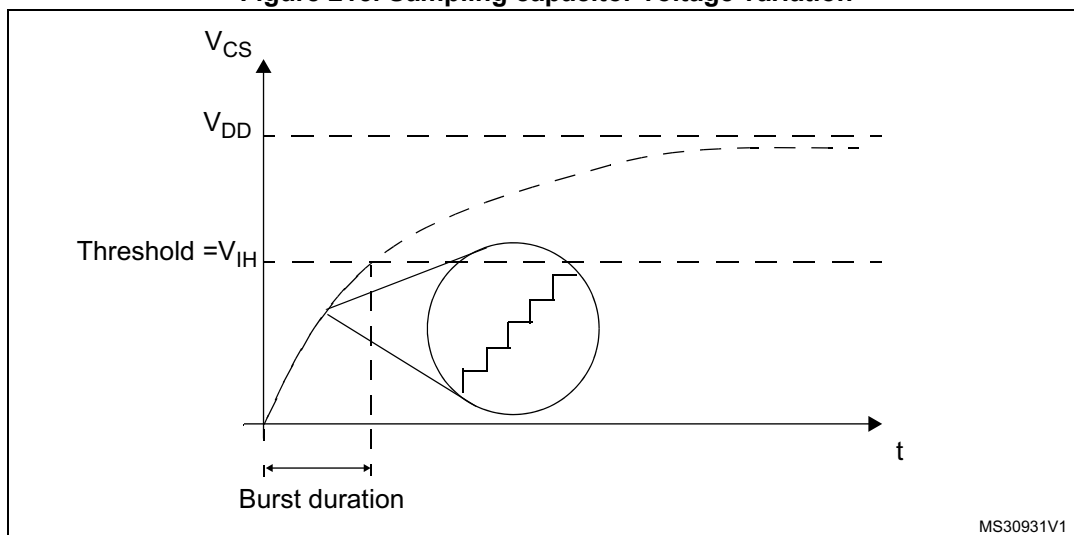
The [Table 188](#) details the charge transfer acquisition sequence of the capacitive sensing channel 1. States 3 to 7 are repeated until the voltage across  $C_S$  reaches the given threshold. The same sequence applies to the acquisition of the other channels. The electrode serial resistor  $R_S$  improves the ESD immunity of the solution.

**Table 188. Acquisition sequence summary**

State	G1_IO1 (channel)	G1_IO2 (sampling)	G1_IO3 (channel)	G1_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

The voltage variation over the time on the sampling capacitor  $C_S$  is detailed below:

**Figure 215. Sampling capacitor voltage variation**



### 30.3.3 Reset and clocks

The TSC clock source is the AHB clock (HCLK). Two programmable prescalers are used to generate the pulse generator and the spread spectrum internal clocks:

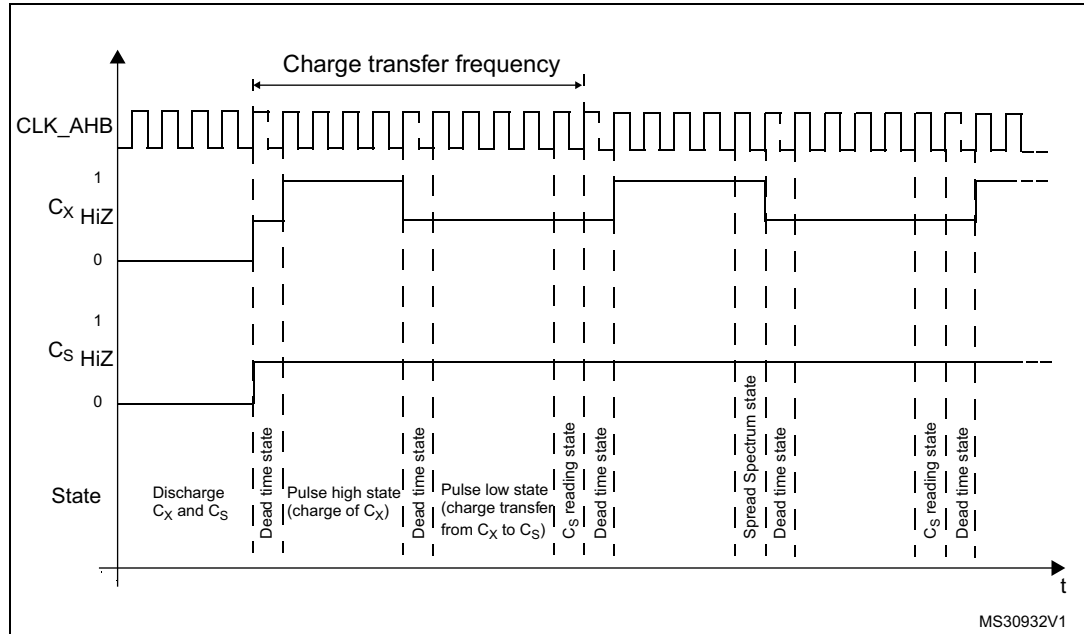
- The pulse generator clock (PGCLK) is defined using the PGPSC[2:0] bits of the TSC\_CR register
- The spread spectrum clock (SSCLK) is defined using the SSPSC bit of the TSC\_CR register

The Reset and Clock Controller (RCC) provides dedicated bits to enable the touch sensing controller clock and to reset this peripheral. For more information, refer to [Section 6: Reset and clock control \(RCC\)](#).

### 30.3.4 Charge transfer acquisition sequence

An example of a charge transfer acquisition sequence is detailed in [Figure 216](#).

**Figure 216. 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 1 periods of HCLK.

At the end of the pulse high state and if the spread spectrum feature is enabled, a variable number of periods of the SSCLK clock are added.

The reading of the sampling capacitor I/O, to determine if the voltage across  $C_S$  has reached the given threshold, is performed at the end of the pulse low state and its duration is one period of HCLK.

*Note:* The following TSC control register configurations are forbidden:

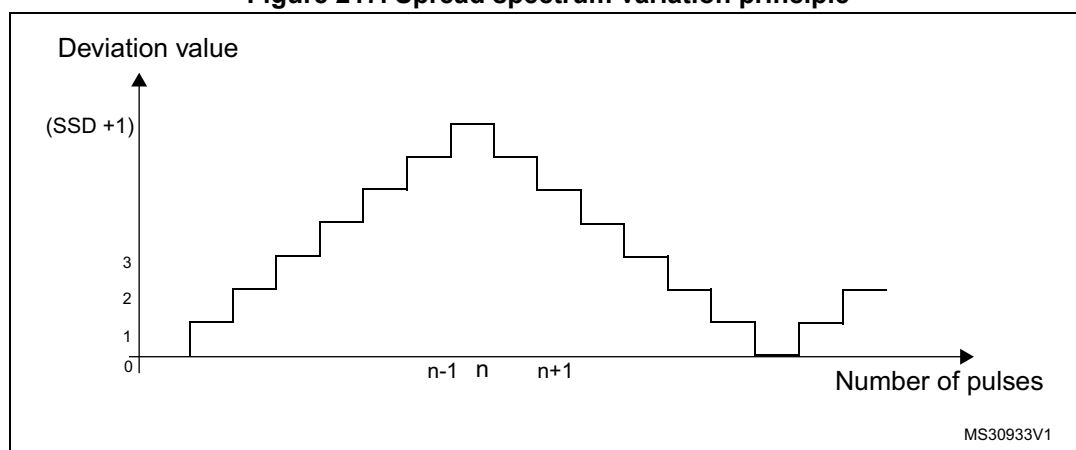
- bits PGPSC are set to '000' and bits CTPL are set to '0000'
- bits PGPSC are set to '000' and bits CTPL are set to '0001'
- bits PGPSC are set to '001' and bits CTPL are set to '0000'

### 30.3.5 Spread spectrum feature

The spread spectrum feature allows to generate a variation of the charge transfer frequency. This is done to improve the robustness of the charge transfer acquisition in noisy environments and also to reduce the induced emission. The maximum frequency variation is in the range of 10% to 50% of the nominal charge transfer period. For instance, for a nominal charge transfer frequency of 250 kHz (4 μs), 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 217. Spread spectrum variation principle



The table below details the maximum frequency deviation with different HCLK settings:

Table 189. Spread spectrum deviation versus AHB clock frequency

f <sub>HCLK</sub>	Spread spectrum step	Maximum spread spectrum deviation
24 MHz	41.6 ns	10666.6 ns
48 MHz	20.8 ns	5333.3 ns
80 MHz	12.5 ns	3205.1 ns
120 MHz	8.3 ns	2128.2 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.

### 30.3.6 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.

### 30.3.7 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 190. 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.*



### 30.3.8 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\_IQGCSR registers specify which analog I/O groups are enabled (corresponding counter is counting). The  $C_S$  voltage of a disabled analog I/O group is not monitored and this group does not participate in the triggering of the end of acquisition flag. However, if the disabled analog I/O group contains some channels, they will be pulsed.

When the  $C_S$  voltage of an enabled analog I/O group reaches the given threshold, the corresponding GxS bit of the TSC\_IQGCSR 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 EOAIIE 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 (EOAIIE and MCEIIE 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 EOAIIC and MCEIIC 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.

### 30.3.9 I/O hysteresis and analog switch control

In order to offer a higher flexibility, the touch sensing controller also allows to take the control of the Schmitt trigger hysteresis and analog switch of each Gx\_I/Oy. This control is available whatever the I/O control mode is (controlled by standard GPIO registers or other peripherals) assuming that the touch sensing controller is enabled. This may be useful to perform a different acquisition sequence or for other purposes.

In order to improve the system immunity, the Schmitt trigger hysteresis of the GPIOs controlled by the TSC must be disabled by resetting the corresponding Gx\_I/Oy bit in the TSC\_IQHCR register.

## 30.4 TSC low-power modes

**Table 191. 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	

## 30.5 TSC interrupts

**Table 192. 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

## 30.6 TSC registers

Refer to [Section 1.2 on page 75](#) of the reference manual for a list of abbreviations used in register descriptions.

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

### 30.6.1 TSC control register (TSC\_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
CTPH[3:0]				CTPL[3:0]				SSD[6:0]								SSE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
SSPSC	PGPSC[2:0]			Res.	Res.	Res.	Res.	MCV[2:0]			IODEF	SYNC POL	AM	START	TSCE	
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:28 **CTPH[3:0]**: Charge transfer pulse high

These bits are set and cleared by software. They define the duration of the high state of the charge transfer pulse (charge of  $C_X$ ).

0000:  $1 \times t_{PGCLK}$

0001:  $2 \times t_{PGCLK}$

...

1111:  $16 \times t_{PGCLK}$

*Note: These bits must not be modified when an acquisition is ongoing.*

Bits 27:24 **CTPL[3:0]**: Charge transfer pulse low

These bits are set and cleared by software. They define the duration of the low state of the charge transfer pulse (transfer of charge from  $C_X$  to  $C_S$ ).

0000:  $1 \times t_{PGCLK}$

0001:  $2 \times t_{PGCLK}$

...

1111:  $16 \times t_{PGCLK}$

*Note: These bits must not be modified when an acquisition is ongoing.*

*Note: Some configurations are forbidden. Refer to the [Section 30.3.4: 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:  $1 \times t_{SSCLK}$

0000001:  $2 \times t_{SSCLK}$

...

1111111:  $128 \times t_{SSCLK}$

*Note: These bits must not be modified when an acquisition is ongoing.*

Bit 16 **SSE**: Spread spectrum enable

This bit is set and cleared by software to enable/disable the spread spectrum feature.

0: Spread spectrum disabled

1: Spread spectrum enabled

*Note: This bit must not be modified when an acquisition is ongoing.*

Bit 15 **SSPSC**: Spread spectrum prescaler

This bit is set and cleared by software. It selects the AHB clock divider used to generate the spread spectrum clock (SSCLK).

0:  $f_{HCLK}$

1:  $f_{HCLK} / 2$

*Note: This bit must not be modified when an acquisition is ongoing.*

Bits 14:12 **PGPSC[2:0]**: Pulse generator prescaler

These bits are set and cleared by software. They select the AHB clock divider used to generate the pulse generator clock (PGCLK).

000:  $f_{HCLK}$

001:  $f_{HCLK} / 2$

010:  $f_{HCLK} / 4$

011:  $f_{HCLK} / 8$

100:  $f_{HCLK} / 16$

101:  $f_{HCLK} / 32$

110:  $f_{HCLK} / 64$

111:  $f_{HCLK} / 128$

*Note: These bits must not be modified when an acquisition is ongoing.*

*Note: Some configurations are forbidden. Refer to the [Section 30.3.4: 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 **TSC**: 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.*

### 30.6.2 TSC interrupt enable register (TSC\_IER)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCEIE**: Max count error interrupt enable

This bit is set and cleared by software to enable/disable the max count error interrupt.

0: Max count error interrupt disabled

1: Max count error interrupt enabled

Bit 0 **EOAIE**: End of acquisition interrupt enable

This bit is set and cleared by software to enable/disable the end of acquisition interrupt.

0: End of acquisition interrupt disabled

1: End of acquisition interrupt enabled

### 30.6.3 TSC interrupt clear register (TSC\_ICR)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCEIC**: Max count error interrupt clear

This bit is set by software to clear the max count error flag and it is cleared by hardware when the flag is reset. Writing a '0' has no effect.

0: No effect

1: Clears the corresponding MCEF of the TSC\_ISR register

Bit 0 **EOAIC**: End of acquisition interrupt clear

This bit is set by software to clear the end of acquisition flag and it is cleared by hardware when the flag is reset. Writing a '0' has no effect.

0: No effect

1: Clears the corresponding EOAF of the TSC\_ISR register

### 30.6.4 TSC interrupt status register (TSC\_ISR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEF	EOAF
														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 EOAI of the TSC\_ICR register.

- 0: Acquisition is ongoing or not started
- 1: Acquisition is complete

### 30.6.5 TSC I/O hysteresis control register (TSC\_IOHCR)

Address offset: 0x10

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **Gx\_IOy**: Gx\_IOy Schmitt trigger hysteresis mode

These bits are set and cleared by software to enable/disable the Gx\_IOy Schmitt trigger hysteresis.

- 0: Gx\_IOy Schmitt trigger hysteresis disabled
- 1: Gx\_IOy Schmitt trigger hysteresis enabled

*Note: These bits control the I/O Schmitt trigger hysteresis whatever the I/O control mode is (even if controlled by standard GPIO registers).*

### 30.6.6 TSC I/O analog switch control register (TSC\_IOASCR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **Gx\_IOy**: Gx\_IOy analog switch enable

These bits are set and cleared by software to enable/disable the Gx\_IOy analog switch.

0: Gx\_IOy analog switch disabled (opened)

1: Gx\_IOy analog switch enabled (closed)

*Note: These bits control the I/O analog switch whatever the I/O control mode is (even if controlled by standard GPIO registers).*

### 30.6.7 TSC I/O sampling control register (TSC\_IOSCR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **Gx\_IOy**: Gx\_IOy sampling mode

These bits are set and cleared by software to configure the Gx\_IOy as a sampling capacitor I/O. Only one I/O per analog I/O group must be defined as sampling capacitor.

0: Gx\_IOy unused

1: Gx\_IOy used as sampling capacitor

*Note: These bits must not be modified when an acquisition is ongoing.*

*During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC\_IOSCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.*



### 30.6.8 TSC I/O channel control register (TSC\_IOCCR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **Gx\_IOy**: Gx\_IOy channel mode

These bits are set and cleared by software to configure the Gx\_IOy as a channel I/O.

0: Gx\_IOy unused

1: Gx\_IOy used as channel

*Note:* These bits must not be modified when an acquisition is ongoing.

*During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC\_IOCCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.*

### 30.6.9 TSC I/O group control status register (TSC\_IOGCSR)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G8S	G7S	G6S	G5S	G4S	G3S	G2S	G1S
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G8E	G7E	G6E	G5E	G4E	G3E	G2E	G1E
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:16 **GxS**: Analog I/O group x status

These bits are set by hardware when the acquisition on the corresponding enabled analog I/O group x is complete. They are cleared by hardware when a new acquisition is started.

0: Acquisition on analog I/O group x is ongoing or not started

1: Acquisition on analog I/O group x is complete

*Note:* When a max count error is detected the remaining GxS bits of the enabled analog I/O groups are not set.

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

Bits 7:0 **GxE**: Analog I/O group x enable

These bits are set and cleared by software to enable/disable the acquisition (counter is counting) on the corresponding analog I/O group x.

0: Acquisition on analog I/O group x disabled

1: Acquisition on analog I/O group x enabled

### 30.6.10 TSC I/O group x counter register (TSC\_I0GxCR)

x represents the analog I/O group number.

Address offset:  $0x30 + 0x04 * x$ , ( $x = 1..8$ )

Reset value: 0x0000 0000

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

30.6.11 TSC register map

Table 193. TSC 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							
0x0000	TSC_CR	CTPH[3:0]			CTPL[3:0]			SSD[6:0]						SSE		SSPSC		PGPSC[2:0]		Res.		Res.		Res.		Res.		MCV [2:0]		IODEF		SYNCPOL		AM		START		TSCE		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0004	TSC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value																																							
0x0008	TSC_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.	Res.	Res.	Res.		
	Reset value																																							
0x000C	TSC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value																																							
0x0010	TSC_IOHCR	G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1							
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0014	Reserved																																							
0x0018	TSC_IOASCR	G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	Reserved																																							
0x0020	TSC_IOSCR	G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	Reserved																																							
0x0028	TSC_IOCRR	G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	Reserved																																							
0x0030	TSC_IQGCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																							
0x0034	TSC_IQG1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																							
0x0038	TSC_IQG2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 193. 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	TSC_I0G3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0040	TSC_I0G4CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0044	TSC_I0G5CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0048	TSC_I0G6CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x004C	TSC_I0G7CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0050	TSC_I0G8CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.

## 31 True random number generator (RNG)

### 31.1 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 validated according to the German AIS-31 standard.

### 31.2 RNG main features

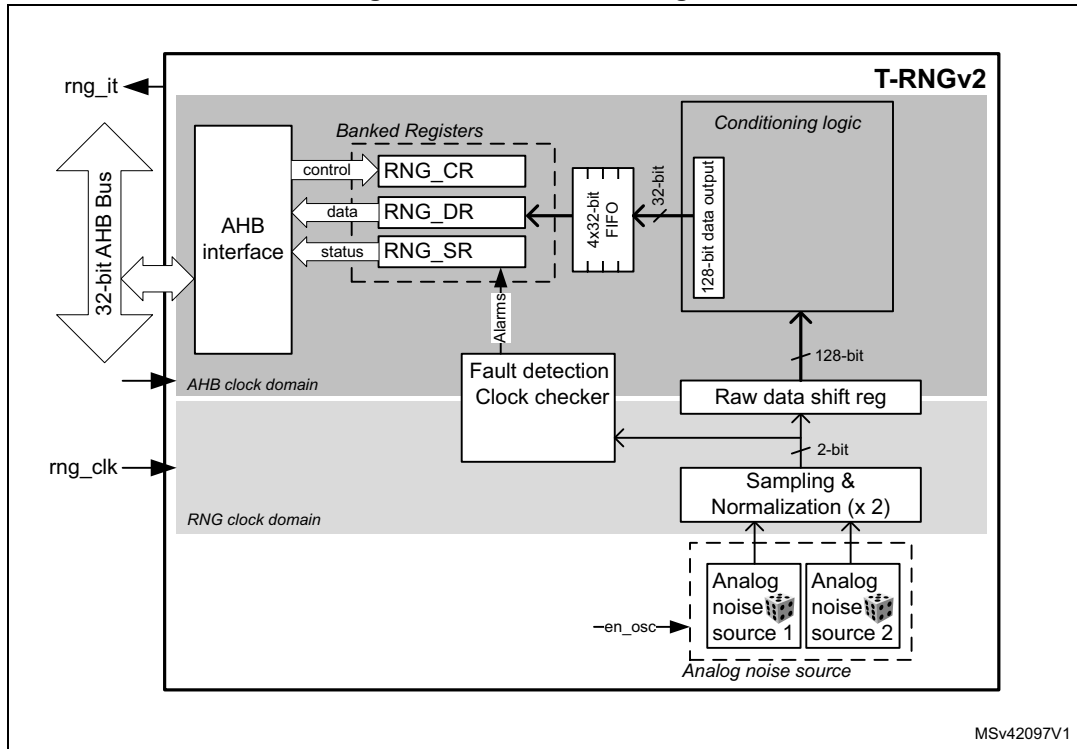
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source processed by a high quality conditioning stage.
- It is validated according to the AIS-31 pre-defined class PTG.2 evaluation methodology, which is part of the German Common Criteria (CC) scheme.
- It produces four 32-bit random samples every  $16 \times \frac{f_{\text{AHB}}}{f_{\text{RNG}}}$  AHB clock cycles, if value is higher than 213 cycles (213 cycles otherwise).
- It allows embedded continuous basic health tests with associated error management
  - Includes too low sampling clock detection and repetition count tests.
- It can be disabled to reduce power consumption.
- It has an AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (else for write accesses an AHB bus error is generated), Warning! any write not equal to 32 bits might corrupt the register content.

### 31.3 RNG functional description

#### 31.3.1 RNG block diagram

Figure 218 shows the RNG block diagram.

Figure 218. RNG block diagram



MSv42097V1

#### 31.3.2 RNG internal signals

Table 194 describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 194. RNG internal input/output signals

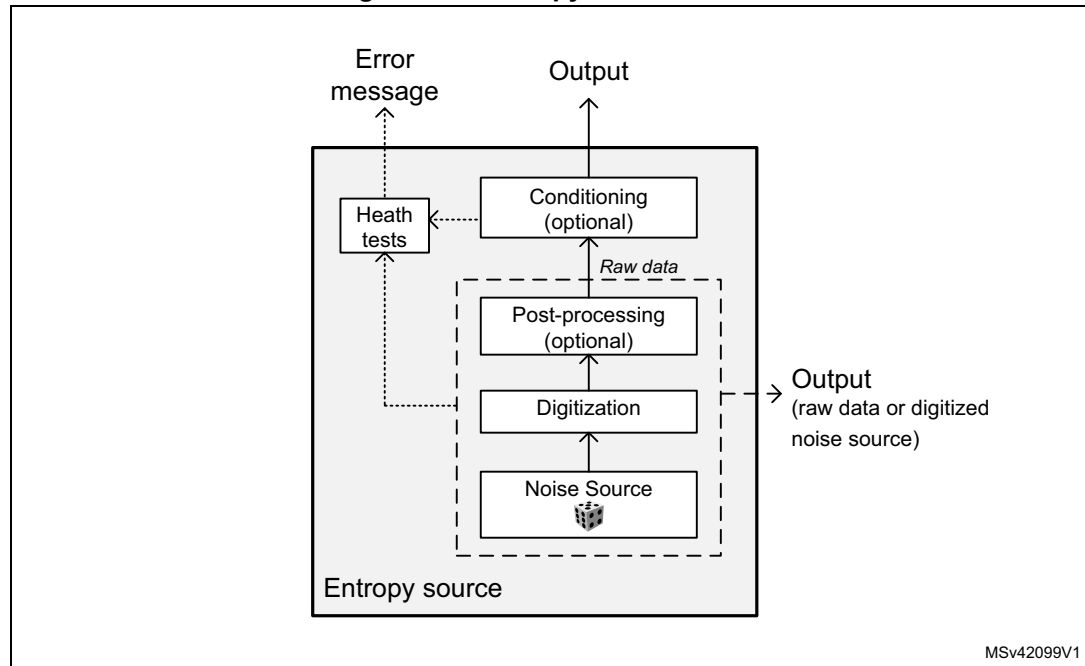
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk2	Digital input	AHB2 clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk2

### 31.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals. The RNG implements the entropy source model pictured on [Figure 219](#), and provides three main functions to the application:

- Collects the bitstring output of the entropy source box
- Obtains samples of the noise source for validation purpose
- Collects error messages from continuous health tests

**Figure 219. Entropy source model**



The main components of the RNG are:

- A source of physical randomness (analog noise sources)
- A digitization stage for those analog noise sources
- A stage delivering a cryptographically conditioned noise source
- Output buffers, for both entropy source output (buffered) and noise source samples (also buffered)
- A health monitoring block performing tests on the whole entropy source

All those components are detailed below.

#### 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 31.4: 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 31.7: 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).

*Note:* The latency during the RNG initialization is described in [Section 31.6: RNG processing time](#).

Also note that post-processing computations are triggered when at least 32 bits of raw data has been 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 31.6: 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. Behavior tests, applied to the entropy source *at run-time*
  - 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 31.3.7: Error management](#).

*Note:* An interrupt can be generated when an error is detected.

### 31.3.4 RNG initialization

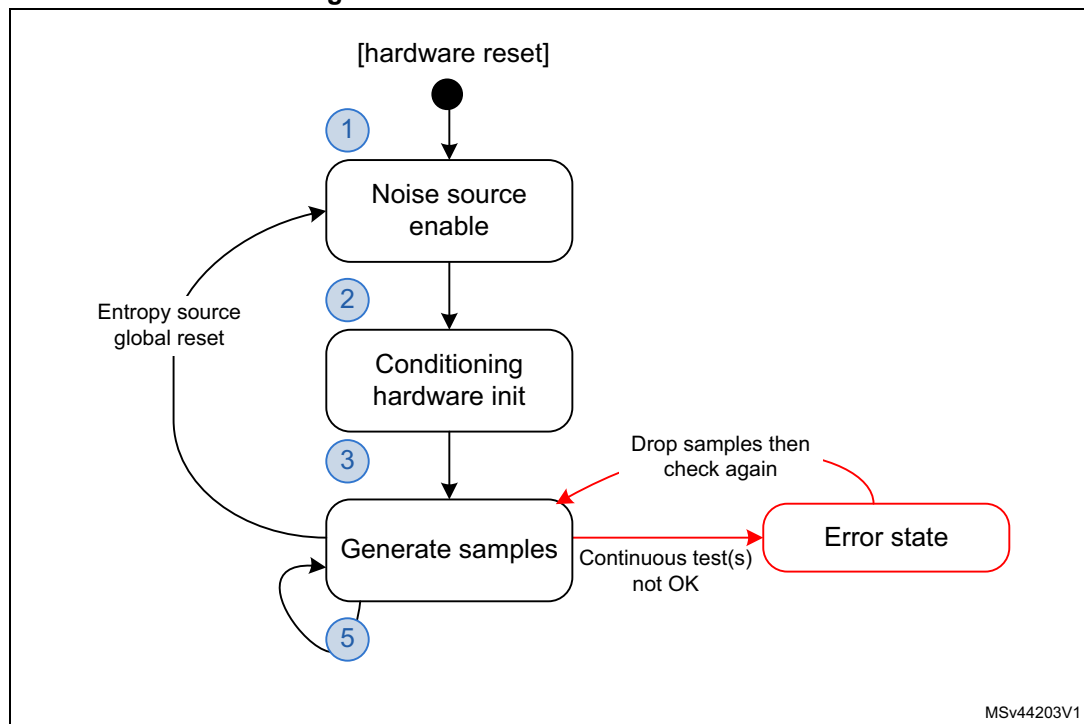
The RNG simplified state machine is pictured on [Figure 220](#)

When a hardware reset occurs 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 31.6: RNG processing time](#).

**Figure 220. RNG initialization overview**



### 31.3.5 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 should 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 will still be 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 31.3.7](#) shall 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 should be set to ‘0’)
  - A random number is ready (the DRDY bit should 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 will still be 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 31.3.7](#) shall be used.

*Note:* When data is not ready (DRDY=“0”) RNG\_DR returns zero.

### Low-power operations

If the power consumption is a concern to the application, low-power strategies can be used, as described in [Section 31.4: RNG low-power usage on page 998](#).

### Software post-processing

No specific software post-processing/conditioning is required to meet AIS-31 approvals. 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 31.3.3: Random number generation](#).

## 31.3.6 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 31.7: Entropy source validation](#).

**Caution:** 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 will flag a clock error (CECS or CEIS in the RNG\_SR register) and the RNG will stop producing random numbers.

See [Section 31.3.1: RNG block diagram](#) for details (AHB and RNG clock domains).

## 31.3.7 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 stops generating random numbers and sets to “1” both the **CEIS** and **CECS** bits to indicate that a clock error occurred. In this case, the application should check that the RNG clock is configured correctly (see [Section 31.3.6: RNG clocking](#)) and then it must clear

the CEIS bit interrupt flag. As soon as the RNG clock operates correctly, the CECS bit will be automatically cleared.

The RNG operates only when the CECS flag is set to “0”. However note that the clock error has no impact on the previously generated random numbers, and the RNG\_DR register contents can still be used.

### 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 will be automatically restarted by the RNG.

The following sequence shall 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.

## 31.4 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 have been collected from the noise source and a complete conditioning round has been 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 (i.e. well before the DRDY bit rises for the first time), the initialization sequence will resume from where it was stopped when RNGEN bit is set to “1”.

## 31.5 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see [Section 31.3.7: Error management](#)
- Clock error, see [Section 31.3.7: Error management](#)

Dedicated interrupt enable control bits are available as shown in [Table 195](#)

**Table 195. RNG interrupt requests**

Interrupt event	Event flag	Enable control bit
Data ready flag	DRDY	IE
Seed error flag	SEIS	IE
Clock error flag	CEIS	IE

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.*

## 31.6 RNG processing time

The conditioning stage can produce four 32-bit random numbers every  $16 \times \frac{f_{\text{AHB}}}{f_{\text{RNG}}}$  clock cycles, if the value is higher than 213 cycles (213 cycles otherwise).

More time is needed for the first set of random numbers after the device exits reset (see [Section 31.3.4: RNG initialization](#)). Indeed, after enabling the RNG for the first time, random data is first available after either:

- 128 RNG clock cycles + 426 AHB cycles, if  $f_{\text{AHB}} < f_{\text{threshold}}$
- 192 RNG clock cycles + 213 AHB cycles, if  $f_{\text{AHB}} \geq f_{\text{threshold}}$

With  $f_{\text{threshold}} = (213 \times f_{\text{RNG}}) / 64$

## 31.7 Entropy source validation

### 31.7.1 Introduction

In order to assess the amount of entropy available from the RNG, STMicroelectronics has tested the peripheral against AIS-31 PTG.2 set of tests. The results can be provided on demand or the customer can reproduce the measurements using the AIS reference software. The customer could also test the RNG against an older NIST SP800-22 set of tests.

### 31.7.2 Validation conditions

STMicroelectronics has validated 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).
- AHB clock rng\_hclk= 216 MHz

### 31.7.3 Data collection

In order to run statistical tests it is required to collect samples from the entropy source at raw data level as well as at the output of the entropy source.

Please contact STMicroelectronics if above samples need to be retrieved for your product.

### 31.8 RNG registers

The RNG is associated with a control register, a data register and a status register.

#### 31.8.1 RNG control register (RNG\_CR)

Address offset: 0x000

Reset value: 0x0000 0000

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

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

Bit 5 **CE**D: Clock error detection

- 0: Clock error detection 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, i.e. 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.

### 31.8.2 RNG status register (RNG\_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY
									rc_w0	rc_w0			r	r	r

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

Bit 6 **SEIS**: Seed error interrupt status

This bit is set at the same time as SECS. It is cleared by writing it to '0'.

0: No faulty sequence detected

1: At least one faulty sequence has been 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 it to '0'.

0: The RNG clock is correct (fRNGCLK > fHCLK/32)

1: The RNG has been 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'.



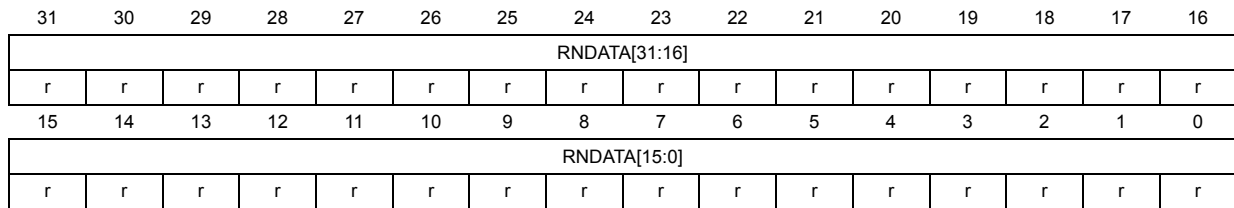
### 31.8.3 RNG data register (RNG\_DR)

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', 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.

### 31.8.4 RNG register map

Table 196 gives the RNG register map and reset values.

Table 196. RNG register map and reset map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RNG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																												0	0	0	0	0
0x004	RNG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	Res.	Res.	Res.	Res.
	Reset value																										0	0			0	0	0
0x008	RNG_DR	RNDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



## 32 AES hardware accelerator (AES)

### 32.1 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.

Multiple chaining modes are supported (ECB, CBC, CTR, GCM, GMAC, CCM), for key sizes of 128 or 256 bits.

The AES accelerator is a 32-bit AHB peripheral. It supports DMA single transfers for incoming and outgoing data (two DMA channels required).

The AES peripheral provides hardware acceleration to AES cryptographic algorithms packaged in STM32 cryptographic library.

AES is an AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated). Other than 32-bit writes may corrupt the register content.

### 32.2 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 key scheduler with its key derivation stage (ECB or CBC decryption only)
- 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

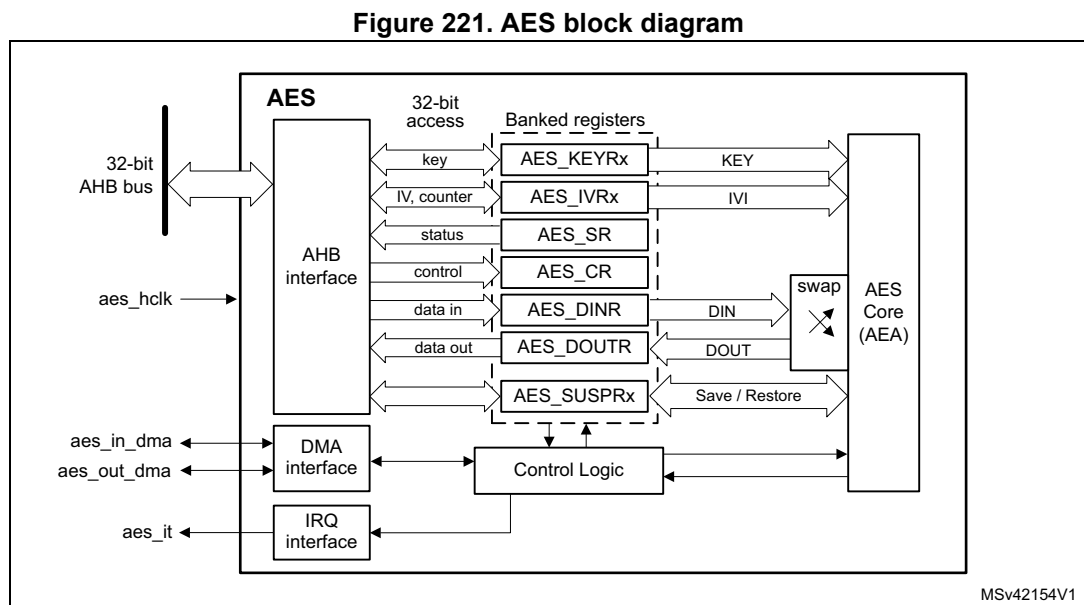
### 32.3 AES implementation

The device has a single instance of AES peripheral.

### 32.4 AES functional description

#### 32.4.1 AES block diagram

Figure 221 shows the block diagram of AES.



#### 32.4.2 AES internal signals

Table 197 describes the user relevant internal signals interfacing the AES peripheral.

**Table 197. AES internal input/output signals**

Signal name	Signal type	Description
aes_hclk	digital input	AHB bus clock
aes_it	digital output	AES interrupt request
aes_in_dma	digital input/output	Input DMA single request/acknowledge
aes_out_dma	digital input/output	Output DMA single request/acknowledge

### 32.4.3 AES cryptographic core

#### Overview

The AES cryptographic core consists of the following components:

- AES 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 96-bit initialization vector IV (and a 32-bit counter field).

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 32.4.4: AES procedure to perform a cipher operation on page 1012](#).

*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)

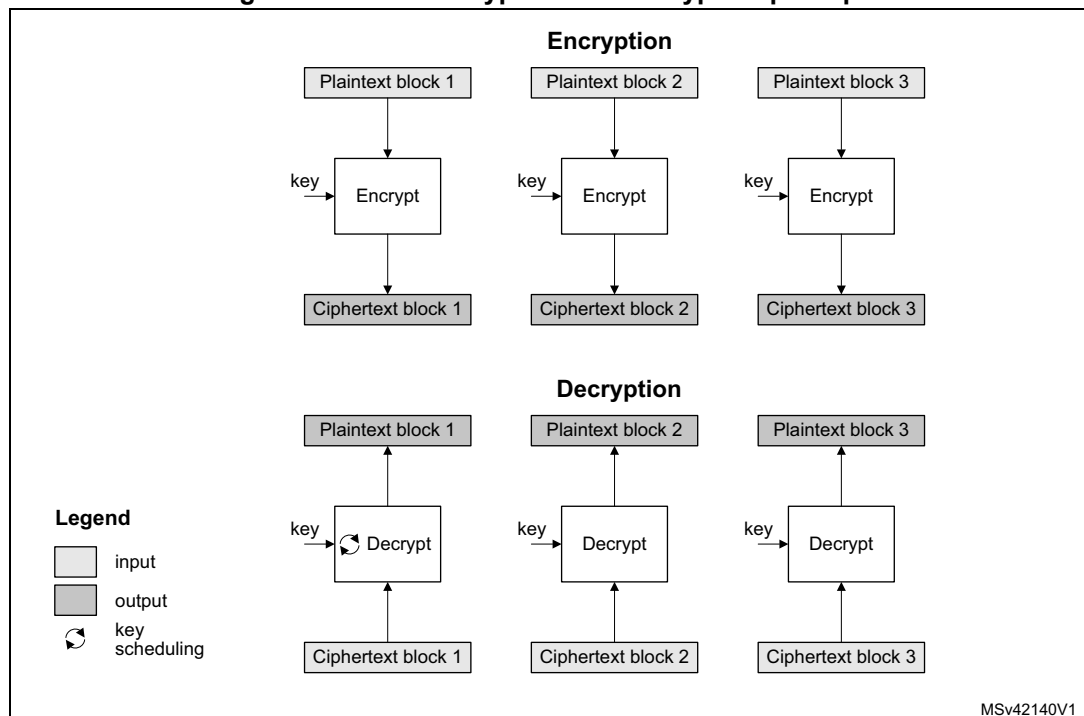
*Note:* The chaining mode may be changed only when AES is disabled (bit EN of the AES\_CR register cleared).

Principle of each AES chaining mode is provided in the following subsections.

Detailed information is in dedicated sections, starting from [Section 32.4.8: AES basic chaining modes \(ECB, CBC\)](#).

### Electronic codebook (ECB) mode

**Figure 222. ECB encryption and decryption principle**

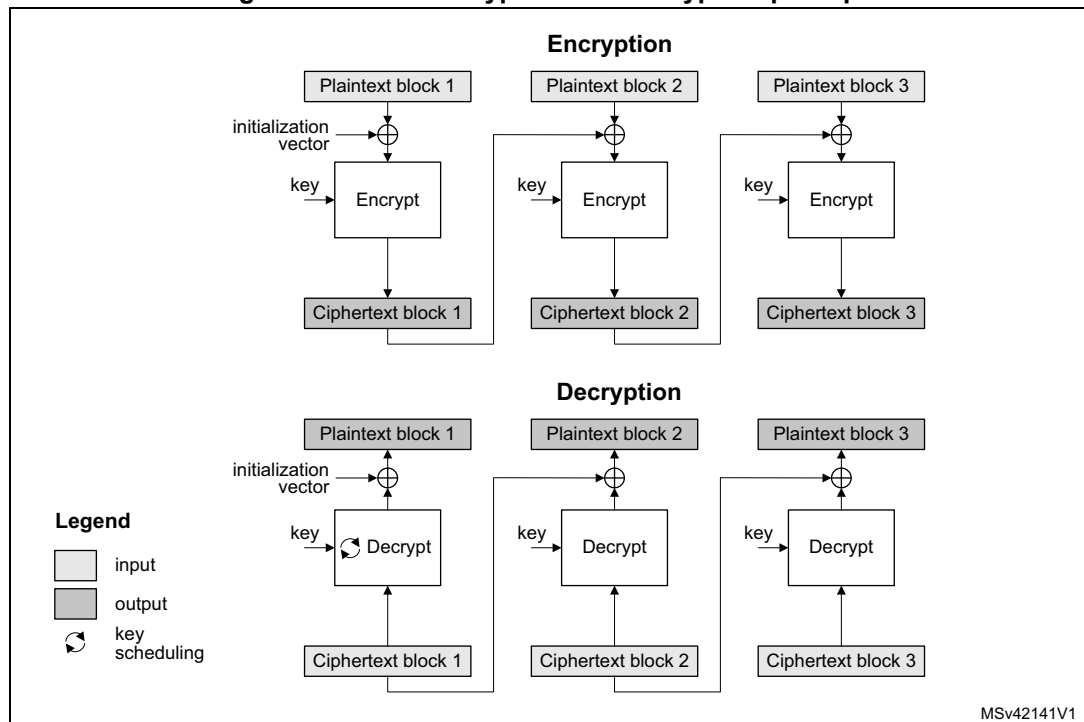


ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately.

*Note:* For decryption, a special key scheduling is required before processing the first block.

Cipher block chaining (CBC) mode

Figure 223. CBC encryption and decryption principle

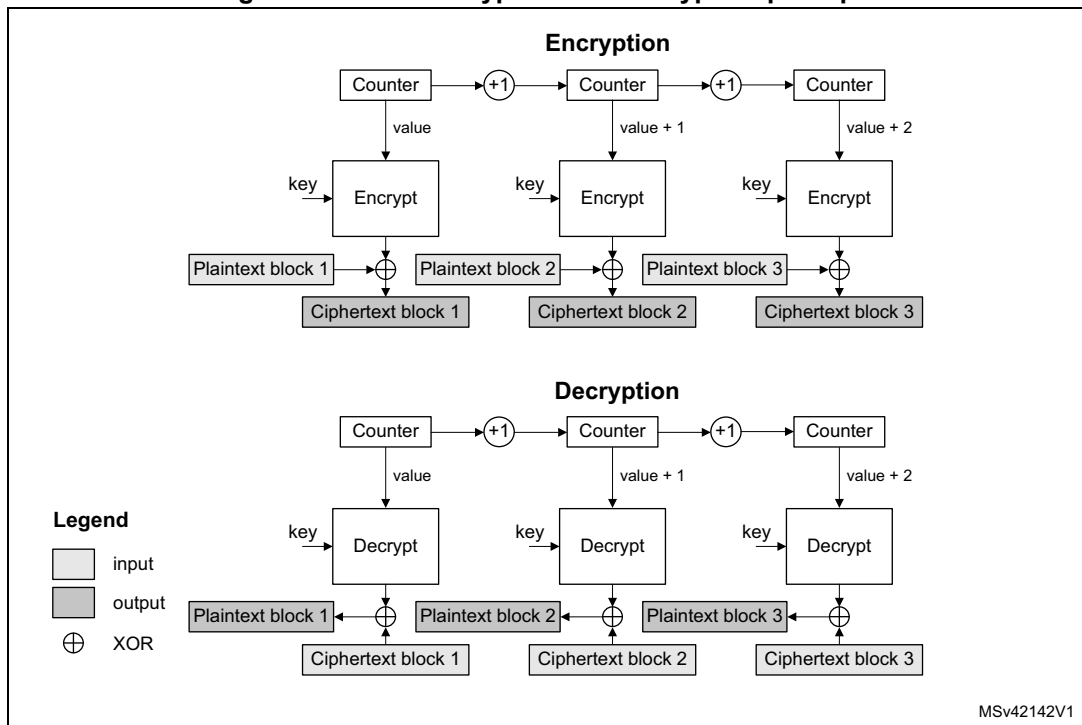


In CBC mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing.

*Note:* For decryption, a special key scheduling is required before processing the first block.

Counter (CTR) mode

Figure 224. CTR encryption and decryption principle



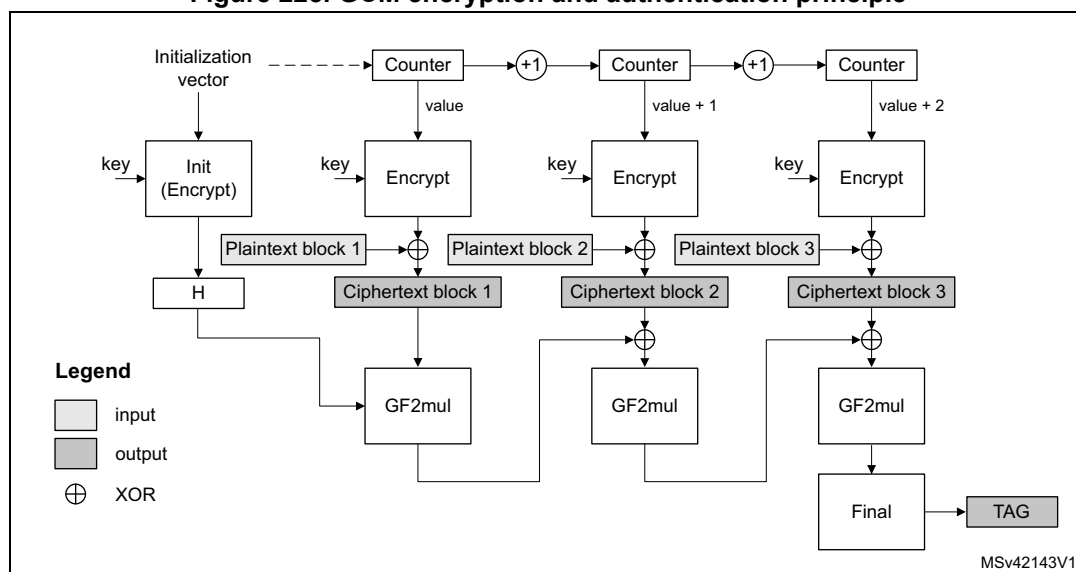
The CTR mode uses the AES core to generate a key stream. The keys are then XORed with the plaintext to obtain the ciphertext as specified in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

*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 225. GCM encryption and authentication principle

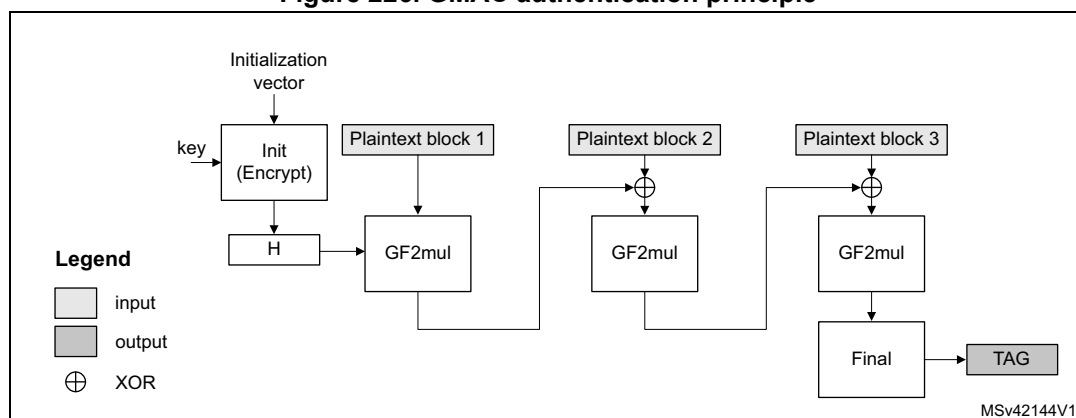


In Galois/counter mode (GCM), the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and its MAC (also known as authentication tag). It is defined in NIST *Special Publication 800-38D, Recommendation for Block 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 226. GMAC authentication principle

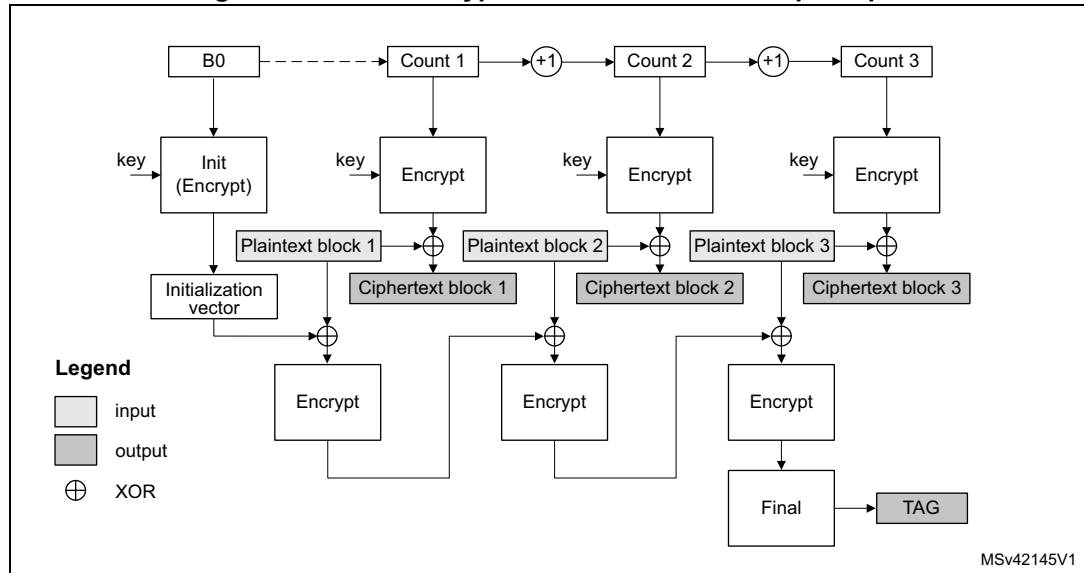


Galois message authentication code (GMAC) allows authenticating a message and generating the corresponding message authentication code (MAC). It is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC*.

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 227. 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 usage is not recommended by NIST.

**32.4.4 AES procedure to perform a cipher operation**

**Introduction**

A typical cipher operation is explained below. Detailed information is provided in sections starting from [Section 32.4.8: AES basic chaining modes \(ECB, CBC\)](#).

The flowcharts shown in [Figure 228](#) and [Figure 229](#) describe the way STM32 cryptographic library implements the AES algorithm. AES accelerates the execution of the AES-128 and AES-256 cryptographic algorithms in ECB, CBC, CTR, CCM, and GCM operating modes.

*Note:* For more details on the cryptographic library, refer to the UM1924 user manual “STM32 crypto library” available from [www.st.com](http://www.st.com).

Figure 228. STM32 cryptolib AES flowchart examples

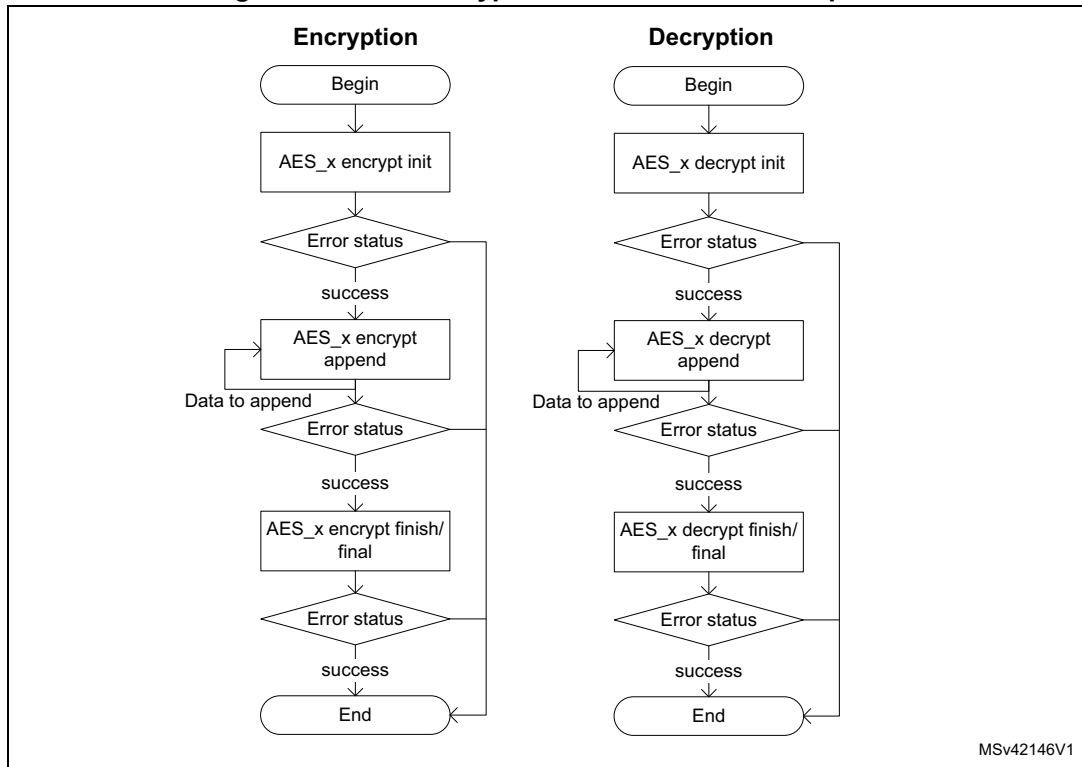
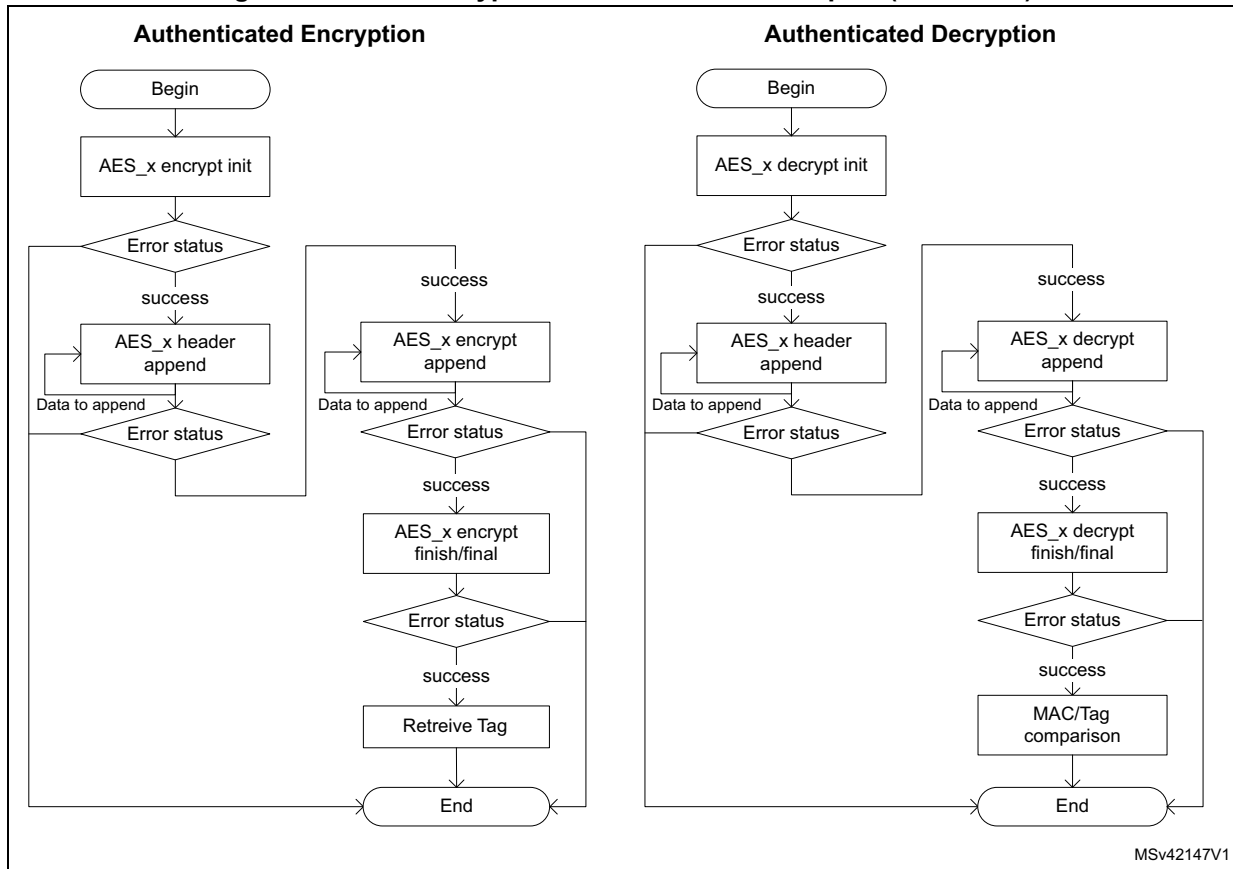


Figure 229. STM32 cryptolib AES flowchart examples (continued)



MSv42147V1

### 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, Mode 1 must be selected (MODE[1:0] = 00).
  - For decryption, Mode 3 must be selected (MODE[1:0] = 10), unless ECB or CBC chaining modes are used. In this latter case, an initial key derivation of the encryption key must be performed, as described in [Section 32.4.5: AES decryption key preparation](#).
- Select the chaining mode, by programming the CHMOD[2:0] bitfield of the AES\_CR register
- 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).
- 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.

### 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 32.4.6: 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. Discard the data that is not part of the payload, then disable the AES peripheral by clearing the EN bit of the AES\_CR register.

*Note:* Up to three wait cycles are automatically inserted between two consecutive writes to the AES\_DINR register, to allow sending the key to the AES processor.

*NPBLB bits are not used in header phase of GCM, GMAC and CCM chaining modes.*

### Data append using interrupt

The method uses interrupt from the AES peripheral to control the data append, through the following sequence:

1. Enable interrupts from AES by setting the CCFIE bit of the AES\_CR register.
2. Enable the AES peripheral by setting the EN bit of the AES\_CR register.
3. Write first four input data words into the AES\_DINR register.
4. Handle the data in the AES interrupt service routine, upon interrupt:
  - a) Read four output data words from the AES\_DOUTR register.
  - b) Clear the CCF flag and thus the pending interrupt, by setting the CCFC bit of the AES\_CR register
  - c) If the data block just processed is the second-last block of an message and the significant data in the last block to process is inferior to 128 bits, pad the remainder of the last block with zeros and, in case of GCM payload encryption or CCM payload decryption, specify the number of non-valid bytes, using the NPBLB

bitfield of the AES\_CR register, for AES to compute a correct tag. Then proceed with point 4e).

- d) If the data block just processed is the last block of the message, discard the data that is not part of the payload, 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 32.4.16: 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 GCM, GMAC and CCM chaining modes.

### 32.4.5 AES decryption key preparation

For an ECB or CBC decryption, a key for the first round of decryption must be derived from the key of the last round of encryption. This is why a complete key schedule of encryption is required before performing the decryption. This key preparation is **not required** for AES decryption in modes other than ECB or CBC.

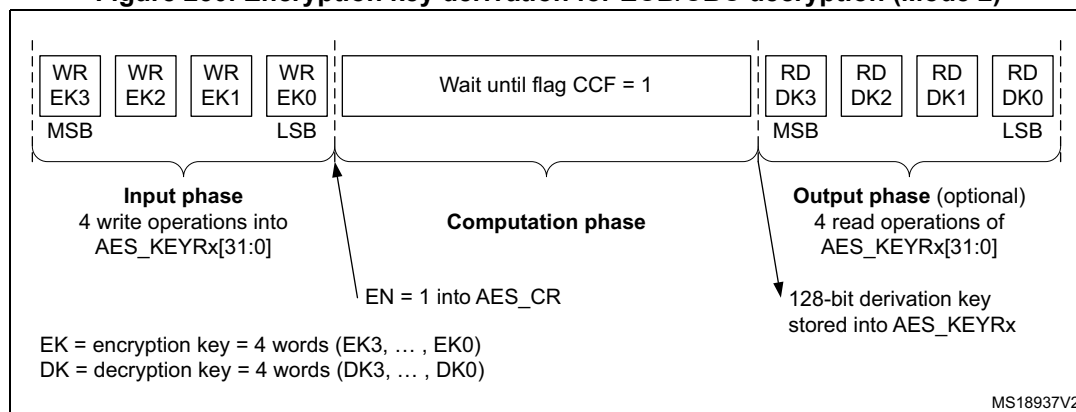
Recommended method is to select the Mode 2 by setting to 01 the MODE[1:0] bitfield of the AES\_CR (key process only), then proceed with the decryption by setting MODE[1:0] to 10 (Mode 3, decryption only). Mode 2 usage is described below:

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 230](#). 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. 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 230](#) (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 230. 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).
- Note:* Alternative key preparation is to select Mode 4 by setting to 11 the MODE[1:0] bitfield of the AES\_CR register. In this case Mode 3 cannot be used.

### 32.4.6 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 32.4.4: AES procedure to perform a cipher operation on page 1012](#) 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 32.4.13: AES data registers and data swapping on page 1039 for details).

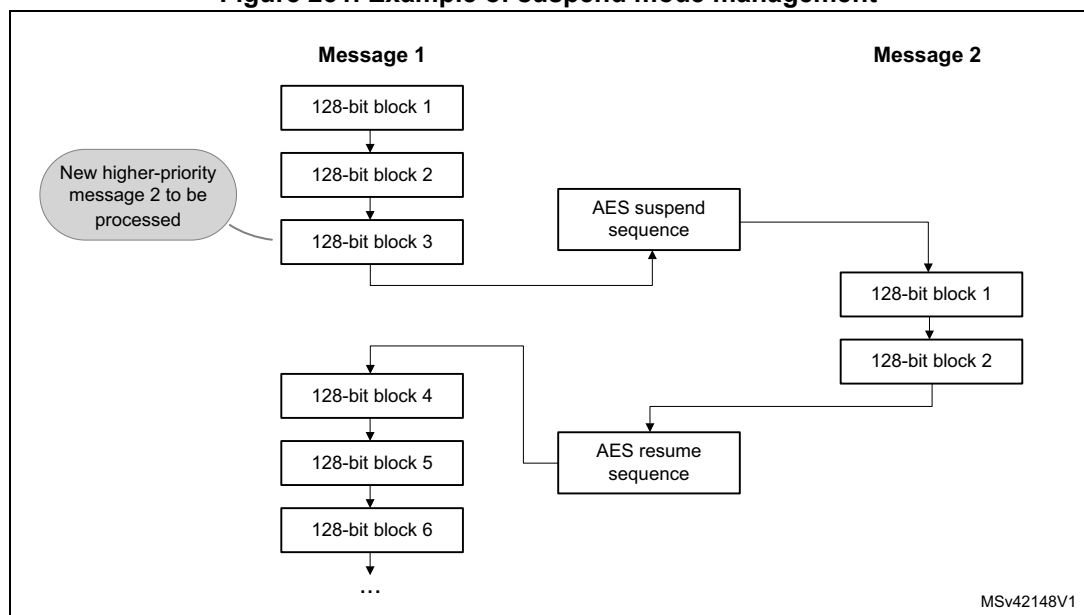
### 32.4.7 AES task suspend and resume

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 231 gives an example of suspend/resume operation: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 231. Example of suspend mode management



A detailed description of suspend/resume operations is in the sections dedicated to each AES mode.

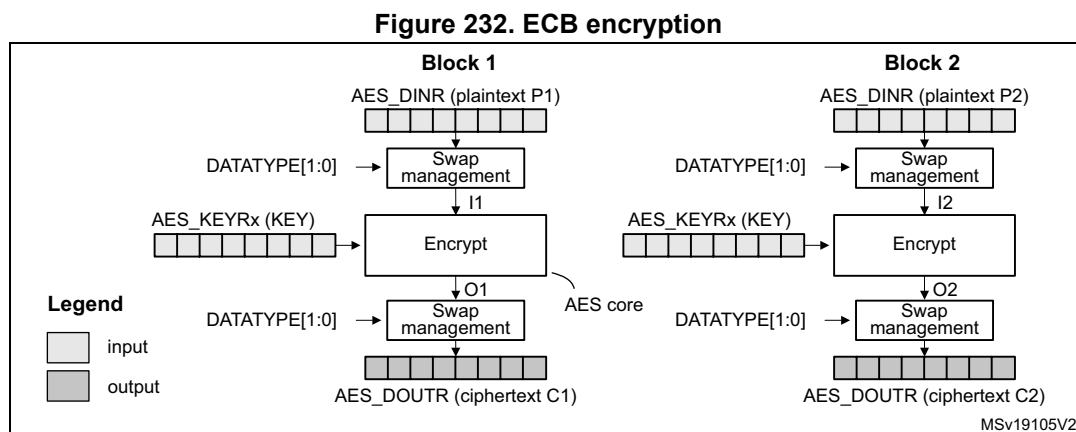
### 32.4.8 AES basic chaining modes (ECB, CBC)

#### Overview

This section gives a brief explanation of the four basic operation modes provided by the AES computing core: ECB encryption, ECB decryption, CBC encryption and CBC decryption. For detailed information, refer to the FIPS publication 197 from November 26, 2001.

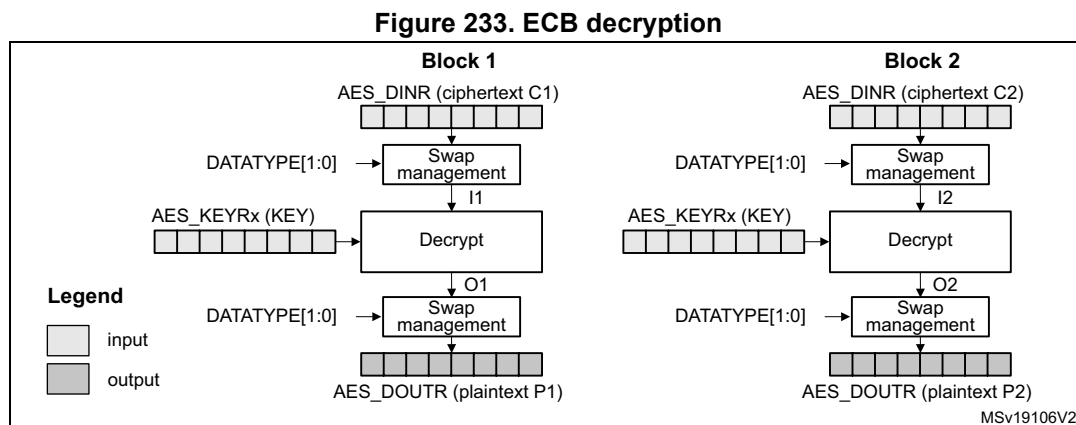


Figure 232 illustrates the electronic codebook (ECB) encryption.



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 O<sub>x</sub> goes through bit/byte/half-word swapping, then is stored in the AES\_DOUTR register as 128-bit ciphertext output data block C<sub>x</sub>. The ECB encryption continues in this way until the last complete plaintext block is encrypted.

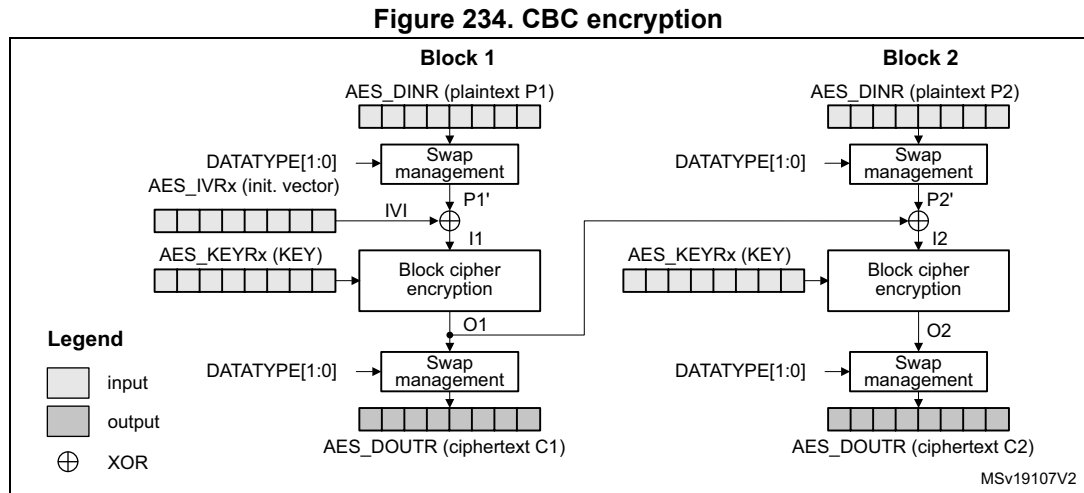
Figure 233 illustrates the electronic codebook (ECB) decryption.



To perform an AES decryption in the ECB mode, the secret key has to be prepared by collecting the last-round encryption key (which requires to first execute the complete key schedule for encryption), and using it as the first-round key for the decryption of the ciphertext. This preparation is supported by the AES core.

In ECB decrypt mode, the 128-bit ciphertext input data block 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.

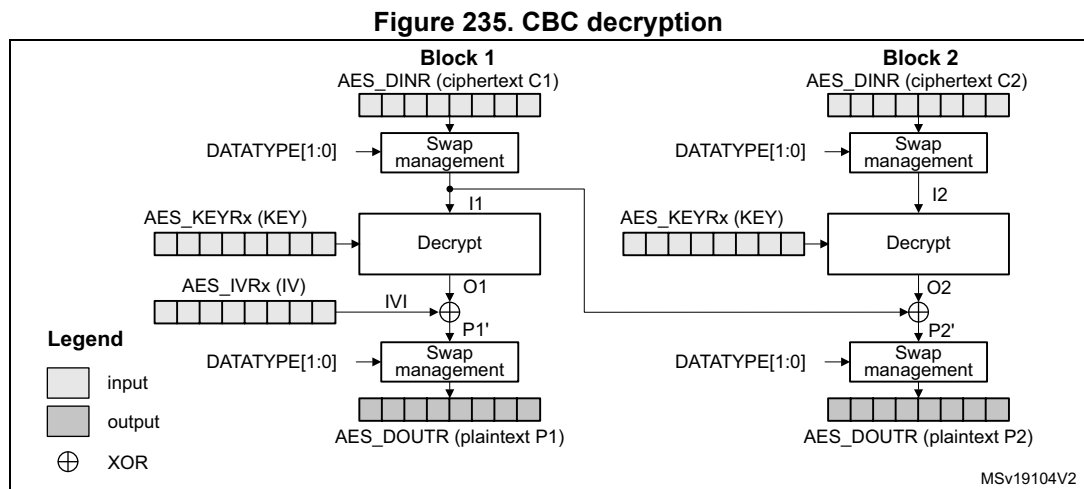
Figure 234 illustrates the cipher block chaining (CBC) encryption mode.



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 32.4.6: AES ciphertext stealing and data padding](#).

Figure 235 illustrates the cipher block chaining (CBC) decryption.



In CBC decrypt mode, like in ECB decrypt mode, the secret key must be prepared to perform an AES decryption.

After the key preparation process, the decryption goes as follows: the first 128-bit ciphertext block (after the swap operation) is used directly as the AES core input block I1 for decrypt operation, using the 128-bit or 256-bit key. Its output O1 is XOR-ed with the 128-bit IVI field (that must be identical to that used during encryption) to produce the first plaintext block P1.

The second ciphertext block is processed in the same way as the first block, except that the 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 32.4.6: AES ciphertext stealing and data padding](#).

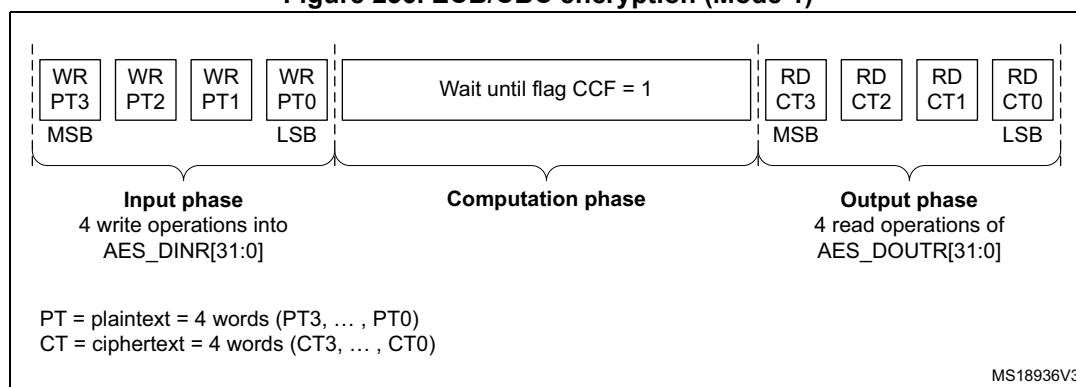
For more information on data swapping, refer to [Section 32.4.13: AES data registers and data swapping](#).

### ECB/CBC encryption sequence

The sequence of events to perform an ECB/CBC encryption (more detail in [Section 32.4.4](#)):

1. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
2. Select the Mode 1 by 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 236](#).
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 236](#). 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 236. ECB/CBC encryption (Mode 1)**



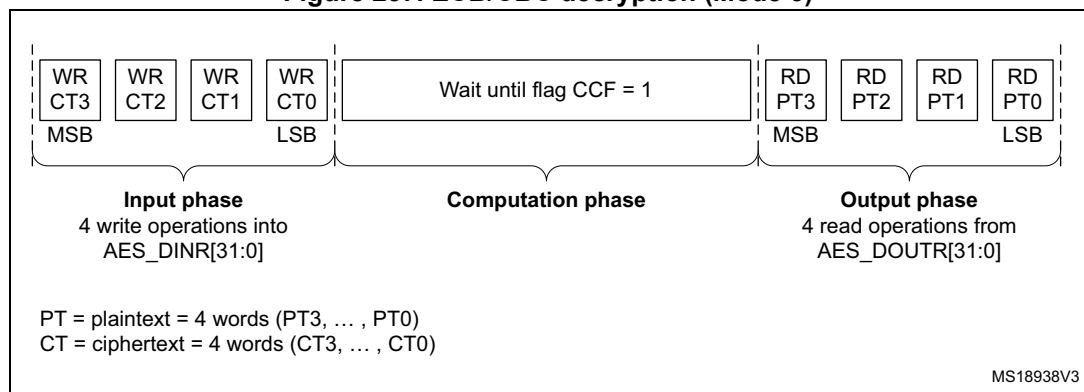
### ECB/CBC decryption sequence

The sequence of events to perform an AES ECB/CBC decryption is as follows (more detail in [Section 32.4.4](#)):

1. Follow the steps described in [Section 32.4.5: AES decryption key preparation on page 1016](#), 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.
3. Select key length of 128 or 256 bits via KEYSIZE bitfield of the AES\_CR register.
  4. Write the AES\_IVRx registers with the initialization vector (required in CBC mode only).
  5. Enable AES 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), as shown in *Figure 237*.
  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), as shown in *Figure 237*. 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 encrypted with the same key.

**Figure 237. 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 current AES configuration in the memory (except AES initialization vector values).
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. Ensure that AES is disabled (the EN bit of the AES\_CR must be 0).
3. Restore the AES\_CR and AES\_KEYRx register setting, using the values of the saved configuration. 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 32.4.5: AES decryption key preparation](#) (only required for ECB or CBC decryption). This step is not necessary if derived key information has been 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 000 or 001, 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.*

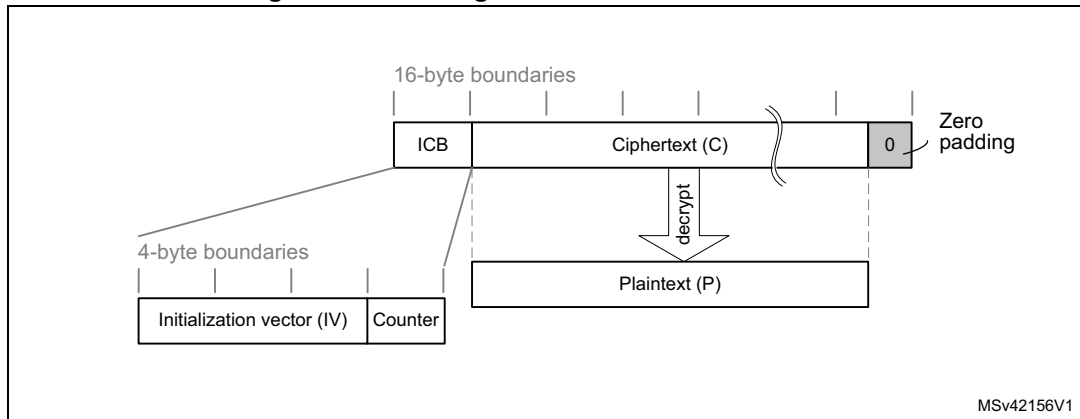
## 32.4.9 AES counter (CTR) mode

### 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 238](#).

Figure 238. 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 should 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 239 and Figure 240 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 239. CTR encryption

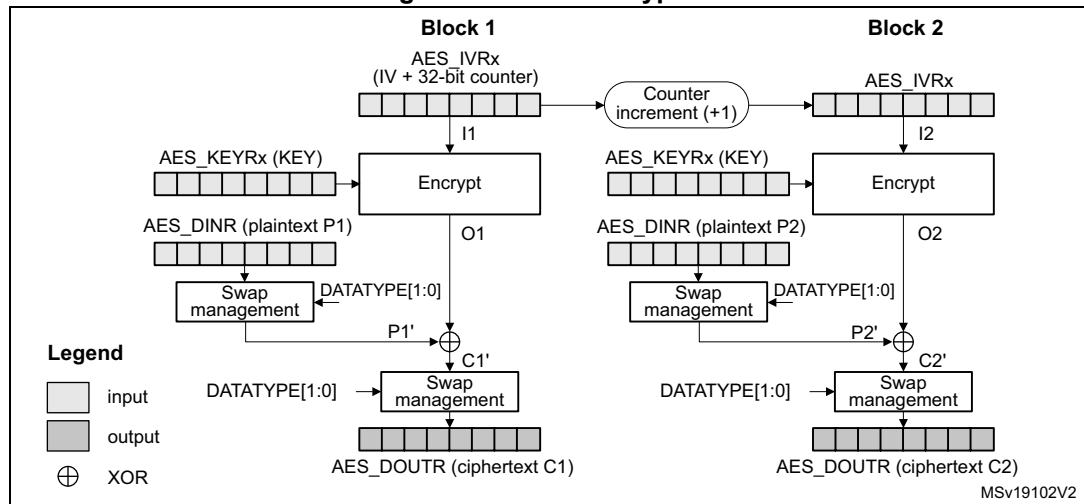
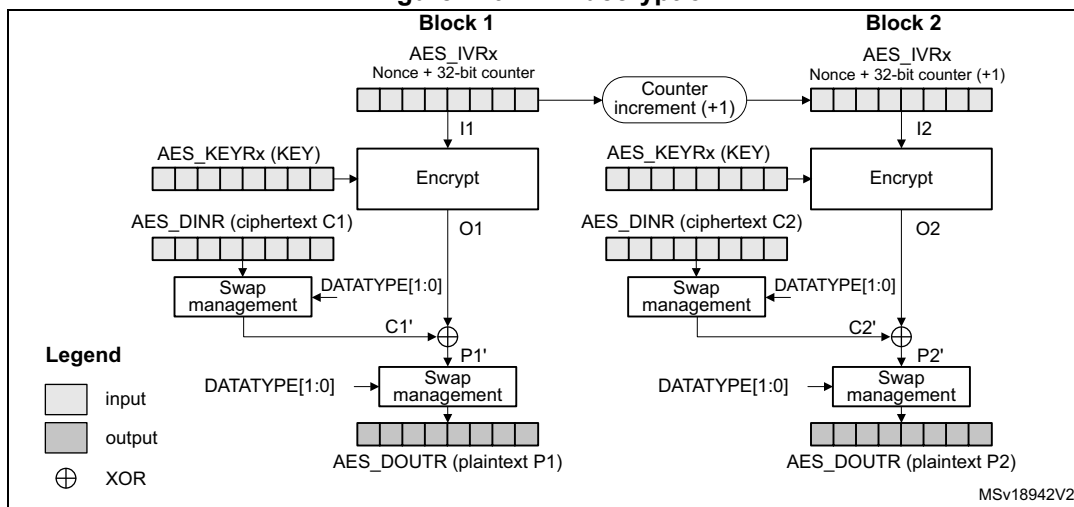


Figure 240. CTR decryption



In CTR mode, the cryptographic core output (also called keystream)  $Ox$  is XOR-ed with relevant input block ( $Px'$  for encryption,  $Cx'$  for decryption), to produce the correct output block ( $Cx'$  for encryption,  $Px'$  for decryption). Initialization vectors in AES must be initialized as shown in [Table 198](#).

Table 198. CTR mode initialization vector definition

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
Nonce[31:0]	Nonce[63:32]	Nonce[95:64]	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 settings 11, 10 or 00 default all to encryption mode, and the setting 01 (key derivation) is forbidden.

The sequence of events to perform an encryption or a decryption in CTR chaining mode:

1. Ensure that AES is disabled (the EN bit of the AES\_CR must be 0).
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 198](#).
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 32.4.4: 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 32.4.8: AES basic chaining modes \(ECB, CBC\)](#).

*Note:* Like for CBC mode, the AES\_IVRx registers must be reloaded during the resume operation.

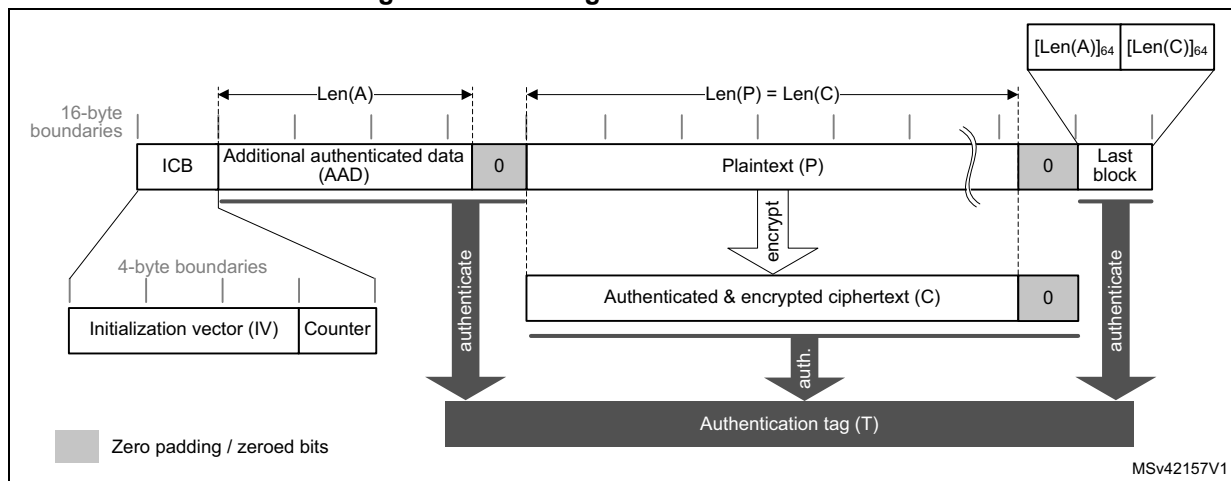
## 32.4.10 AES Galois/counter mode (GCM)

### 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 241](#).

**Figure 241. Message construction in GCM**





The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV)**: a 96-bit value that must be unique for each encryption cycle with a given key. Note that the GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also known as additional authentication data) has a known length  $Len(A)$  that may be a non-multiple of 16 bytes, and must not exceed  $2^{64} - 1$  bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length  $Len(P)$  that may be non-multiple of 16 bytes, and cannot exceed  $2^{32} - 2$  128-bit blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in [Table 199](#).

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.

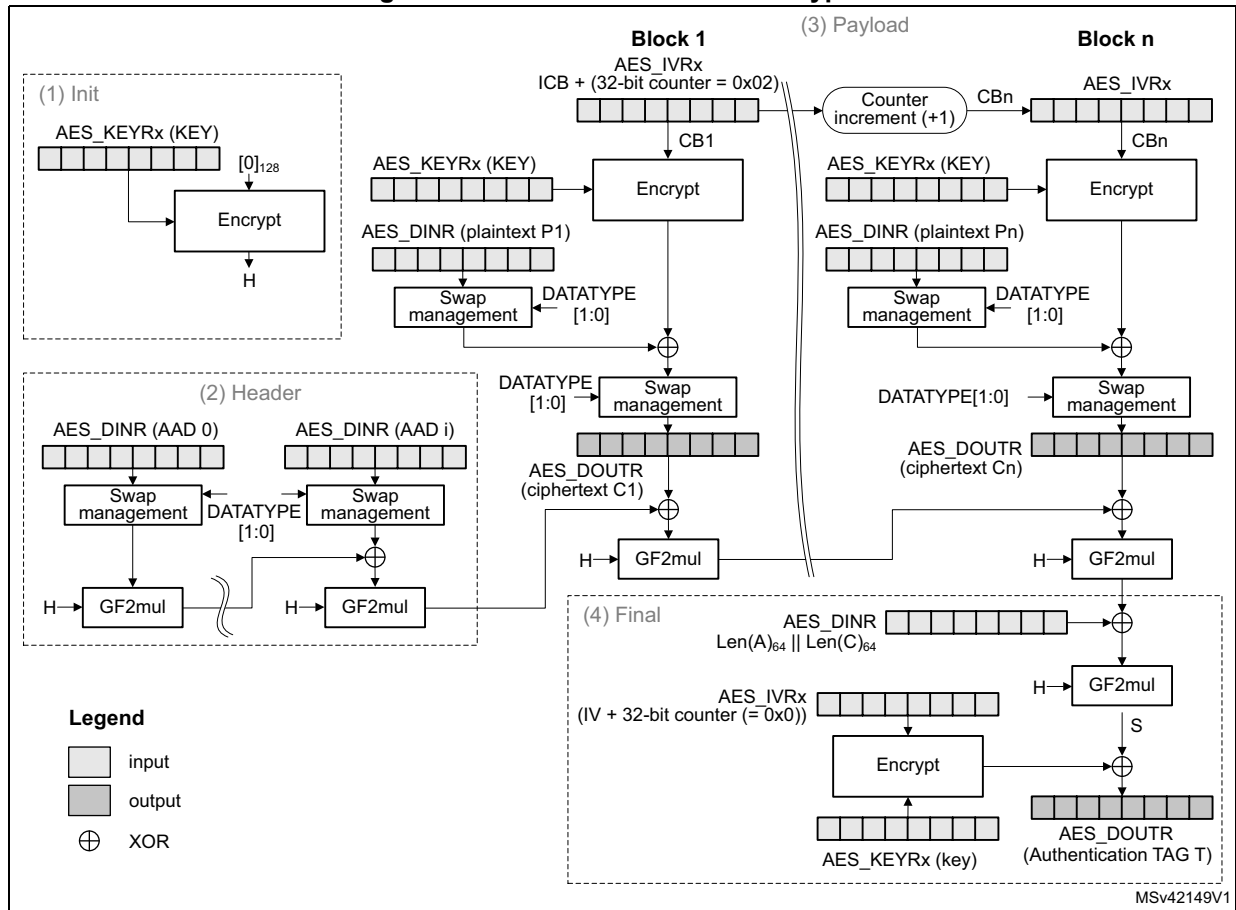
**Table 199. GCM last block definition**

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 242 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 242. 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 200).

Table 200. GCM mode IVI bitfield initialization

Register	AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
Input data	ICB[31:0]	ICB[63:32]	ICB[95:64]	Counter[31:0] = 0x2

Note: In GCM 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. Ensure that AES is disabled (the EN bit of the AES\_CR must be 0).
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.
5. Initialize the AES\_KEYRx registers with a key, and initialize AES\_IVRx registers with the information as defined in [Table 200](#).
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 to 1 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 32.4.4: AES procedure to perform a cipher operation on page 1012](#).
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. If the header phase was skipped, enable the AES peripheral by setting the EN bit of the AES\_CR register.
2. 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.
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 32.4.4: AES procedure to perform a cipher operation on page 1012](#), 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. Select encrypt mode by setting to 00 the MODE[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 199](#). 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 in the AES\_SR register, by setting to 1 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 must be inserted normally (no swapping).

*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 to 1 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. Ensure that the AES peripheral is disabled (the EN bit of the AES\_CR register must be 0).
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.
7. 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.

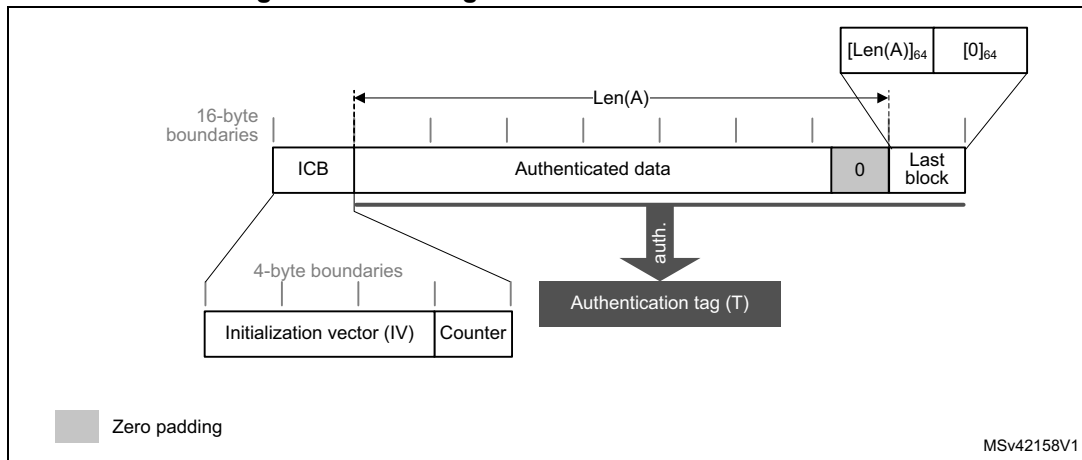
### 32.4.11 AES Galois message authentication code (GMAC)

#### 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 243](#).

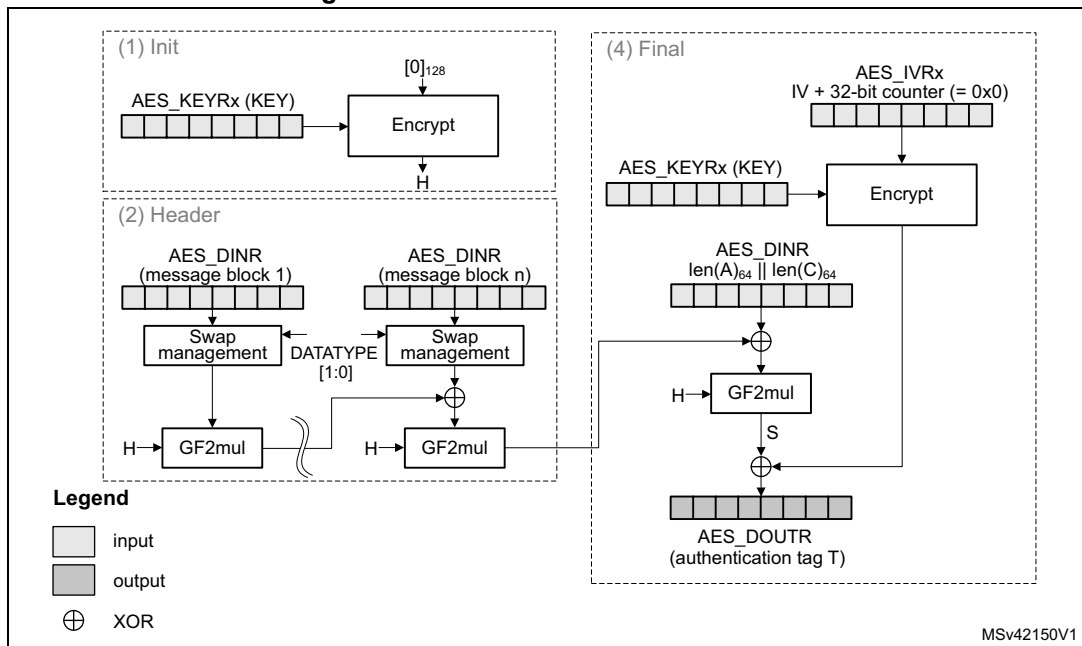
**Figure 243. Message construction in GMAC mode**



**AES GMAC processing**

[Figure 244](#) 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 244. 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.

### 32.4.12 AES counter with CBC-MAC (CCM)

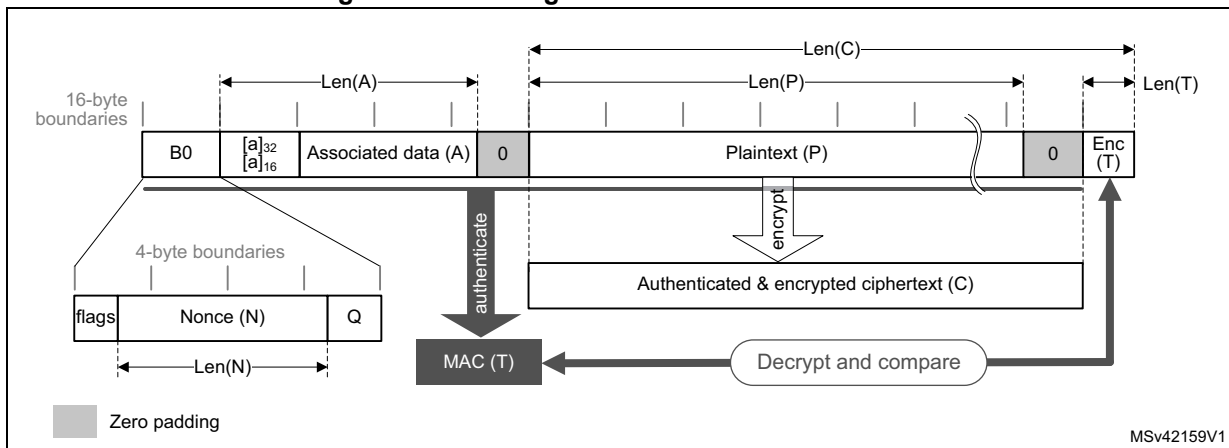
#### Overview

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 245](#).

**Figure 245. 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 should be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags**: most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) < 2<sup>8Q</sup> bytes). The counter blocks range associated to **Q** is equal to 2<sup>8Q-4</sup>, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher shall 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 Len(A) that can be a non-multiple of 16 bytes (see [Figure 245](#)). 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  $0xff || 0xfe || [a]_{32}$ , that is, on six bytes.
  - If  $2^{32} < a < 2^{64}$ , then it is encoded as  $0xff || 0xff || [a]_{64}$ , 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  $Len(P)$ . This length can be a non-multiple of 16 bytes (see [Figure 245](#)).
  - **Encrypted MAC (T)** of length  $Len(T)$  appended to the ciphertext C of overall length  $Len(C)$ .

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

*Note: CCM chaining mode can also be used with associated data only (that is, no payload).*

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

```
N: 10111213 141516 (Len(N)= 56 bits or 7 bytes)
A: 00010203 04050607 (Len(A)= 64 bits or 8 bytes)
P: 20212223 (Len(P)= 32 bits or 4 bytes)
T: 6084341B (Len(T)= 32 bits or t = 4)
B0: 4F101112 13141516 00000000 00000004
B1: 00080001 02030405 06070000 00000000
B2: 20212223 00000000 00000000 00000000
CTR0: 0710111213 141516 00000000 00000000
CTR1: 0710111213 141516 00000000 00000001
```

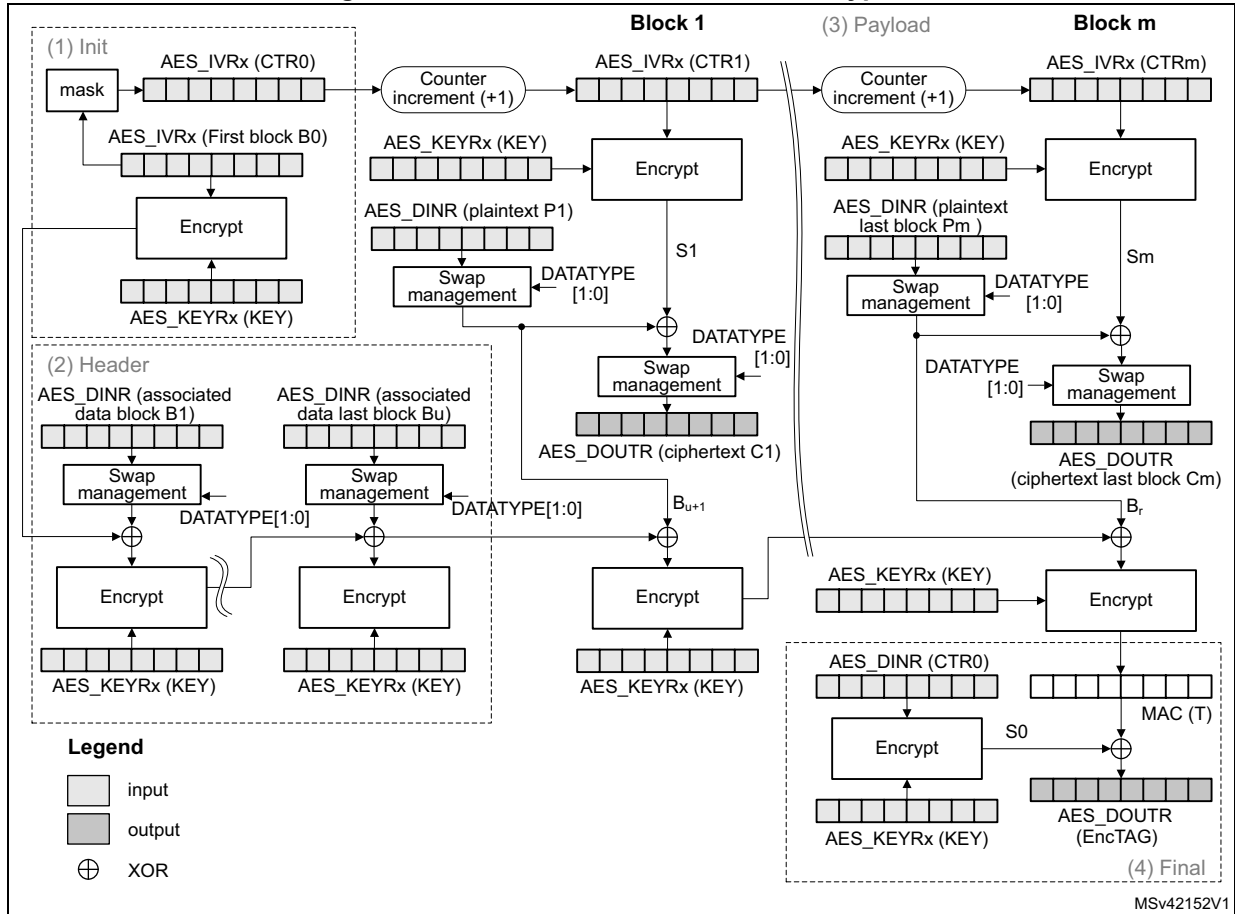
Generation of formatted input data blocks Bx (especially B0 and B1) must be managed by the application.



CCM processing

Figure 246 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 246. 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 201 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 201. Initialization of AES\_IVRx registers in CCM mode**

Register	AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
Input data	B0[31:0]	B0[63:32]	B0[95:64]	B0[127:96]

*Note:* In CCM mode the MODE[1:0] bitfield settings 01 and 11 (key derivation) 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. Ensure that the AES peripheral is disabled (the EN bit of the AES\_CR must be 0).
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.
5. Initialize the AES\_KEYRx registers with a key, and initialize AES\_IVRx registers with B0 data as described in [Table 201](#).
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 32.4.4: AES procedure to perform a cipher operation on page 1012](#).
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 PH[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 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 32.4.4: AES procedure to perform a cipher operation on page 1012](#), 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  $Len(P) = Len(T)$ .

Remove  $LSB_{Len(T)}(C)$  encrypted tag information when decrypting ciphertext C.

#### **CCM final phase**

In this last phase, the AES peripheral generates the GCM authentication tag and stores it in the AES\_DOUTR register. The sequence to execute is:

1. Indicate the final phase, by setting to 11 the GCM PH[1:0] bitfield of the AES\_CR register. Select encrypt mode by keeping 00 in the MODE[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, data is read normally (no swapping).

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. Ensure that the AES peripheral is disabled (the EN bit of the AES\_CR register must be 0).
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.

### 32.4.13 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[127:0]), 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[127:0]), 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 32.4.16: AES DMA interface](#).

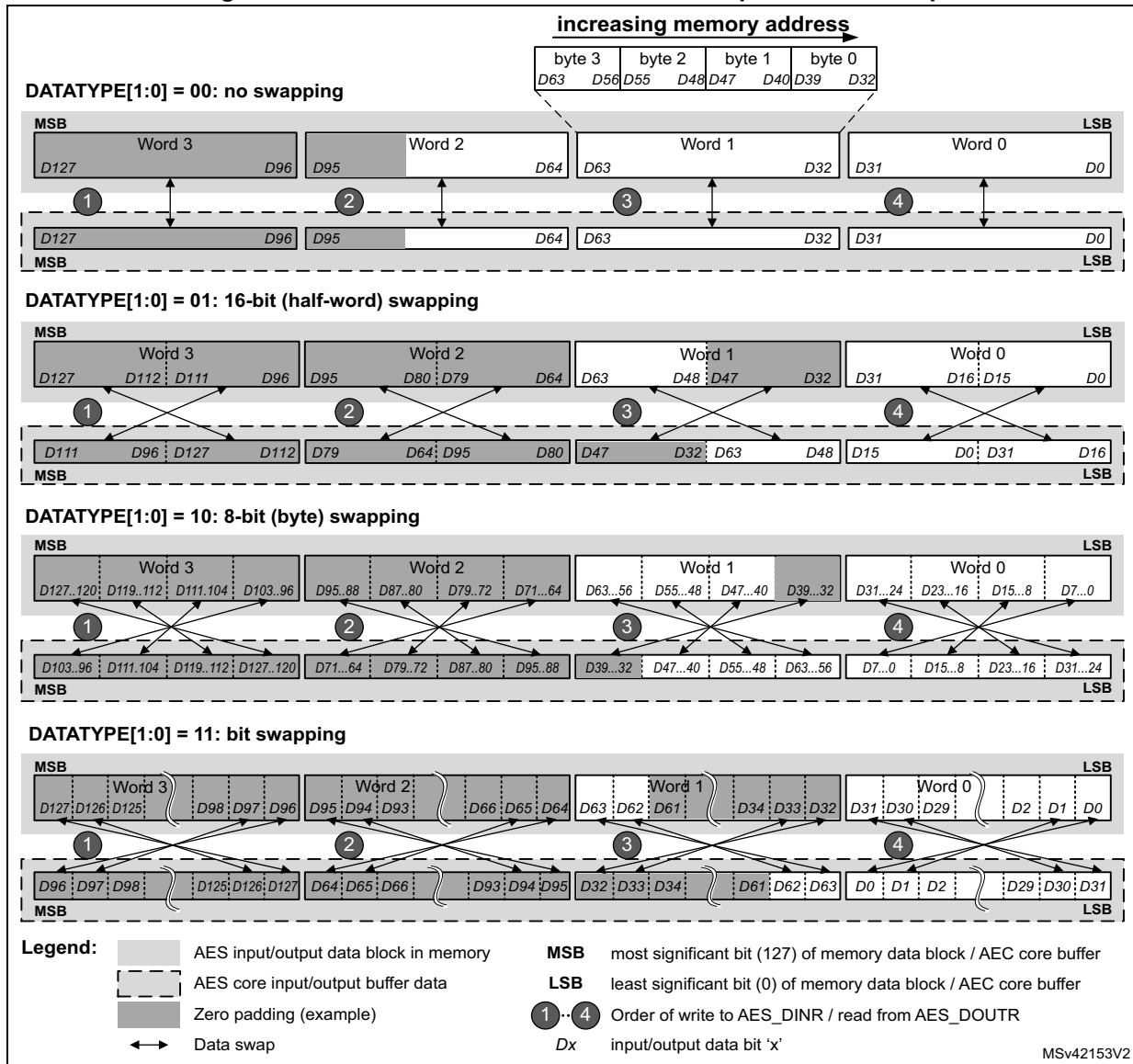
#### Data swapping

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 247](#) shows the construction of AES processing core input buffer data P127..0, 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..0.

Figure 247. 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 247 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

### 32.4.14 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 the eight registers as shown in [Table 202](#).

**Table 202. 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.

The key registers are not affected by the data swapping controlled by DATATYPE[1:0] bitfield of the AES\_CR register.

### 32.4.15 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.

### 32.4.16 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

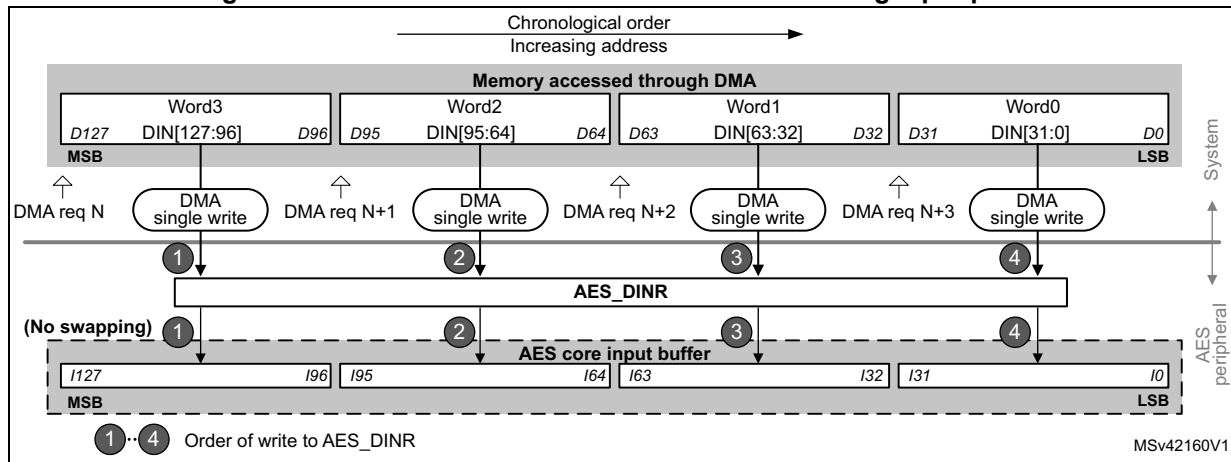
Setting the DMAINEN bit of the AES\_CR register enables DMA writing into AES. The AES peripheral then initiates a DMA request during the input phase each time it requires a word to be written to the AES\_DINR register. It asserts four DMA requests to transfer one 128-bit (four-word) input data block from memory, as shown in [Figure 248](#).

See [Table 203](#) for recommended DMA configuration.

**Table 203. DMA channel configuration for memory-to-AES data transfer**

DMA channel control register field	Recommended configuration
Transfer size	Message length: a multiple of 128 bits. 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. Refer to <a href="#">Section 32.4.4: AES procedure to perform a cipher operation</a> for details.
Source burst size (memory)	Single
Destination burst size (peripheral)	Single
DMA FIFO size	AES FIFO_size = 4 bytes.
Source transfer width (memory)	32-bit words
Destination transfer width (peripheral)	32-bit words
Source address increment (memory)	Yes, after each 32-bit transfer
Destination address increment (peripheral)	Fixed address of AES_DINR (no increment)

**Figure 248. 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 a word to be read from the AES\_DOUTR register. It asserts four DMA requests to transfer one 128-bit (four-word) output data block to memory, as shown in [Figure 249](#).

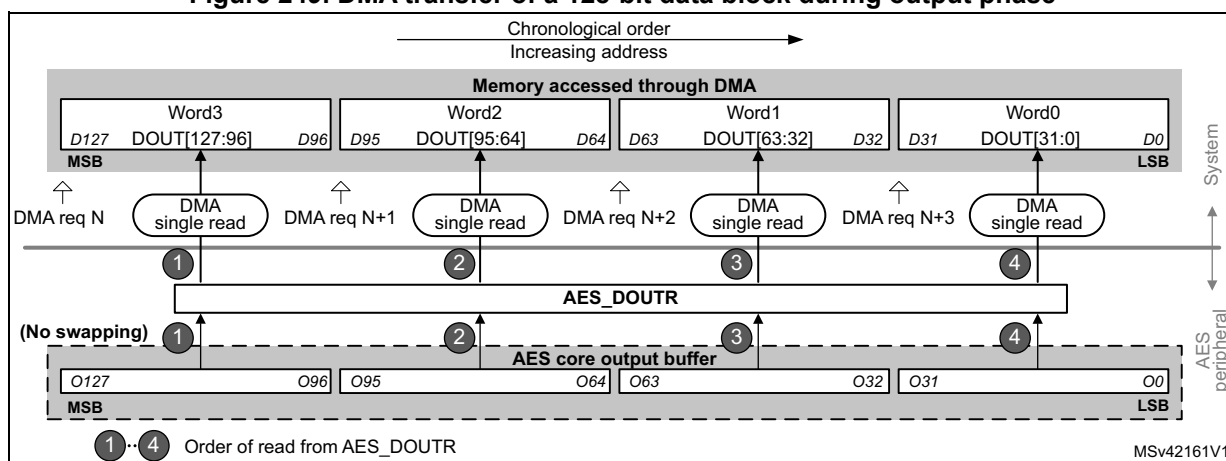
See [Table 204](#) for recommended DMA configuration.



**Table 204. DMA channel configuration for AES-to-memory data transfer**

DMA channel control register field	Recommended configuration
Transfer size	It is the message length multiple of AES block size (4 words). According to the case extra bytes will have to be discarded.
Source burst size (peripheral)	Single
Destination burst size (memory)	Single
DMA FIFO size	AES FIFO_size = 4 bytes
Source transfer width (peripheral)	32-bit words
Destination transfer width (memory)	32-bit words
Source address increment (peripheral)	Fixed address of AES_DINR (no increment)
Destination address increment (memory)	Yes, after each 32-bit transfer

**Figure 249. 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 is not relevant and can be ignored (left set) by software. It must only be cleared when

transiting back to data transferring managed by software. See [Suspend/resume operations in ECB/CBC modes](#) in [Section 32.4.8: AES basic chaining modes \(ECB, CBC\)](#) as example.

### 32.4.17 AES error management

The read error flag (RDERR) and write error flag (WRERR) of the AES\_SR register are set when an unexpected read or write operation, respectively, is detected. An interrupt can be generated if the error interrupt enable (ERRIE) bit of the AES\_CR register is set. For more details, refer to [Section 32.5: AES interrupts](#).

*Note:* AES is not disabled after an error detection and continues processing.

AES can be re-initialized at any moment by clearing then setting the EN bit of the AES\_CR register.

#### Read error flag (RDERR)

When an unexpected read operation is detected during the computation phase or during the input phase, the AES read error flag (RDERR) is set in the AES\_SR register. An interrupt is generated if the ERRIE bit of the AES\_CR register is set.

The RDERR flag is cleared by setting the corresponding ERRC bit of the AES\_CR register.

#### Write error flag (WDERR)

When an unexpected write operation is detected during the computation phase or during the output phase, the AES write error flag (WRERR) is set in the AES\_SR register. An interrupt is generated if the ERRIE bit of the AES\_CR register is set.

The WDERR flag is cleared by setting the corresponding ERRC bit of the AES\_CR register.

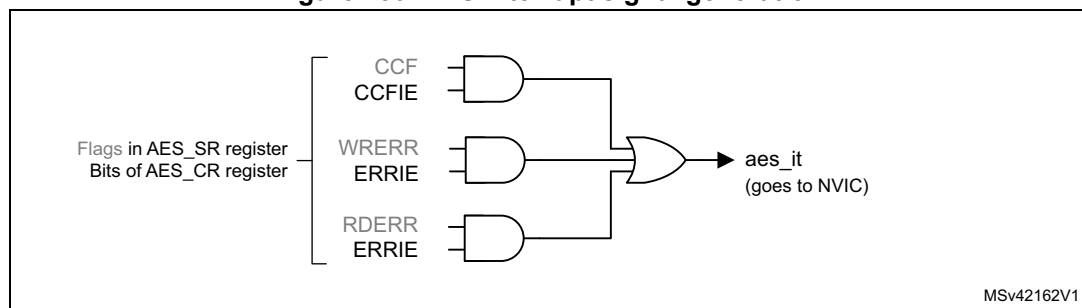
## 32.5 AES interrupts

There are three individual maskable interrupt sources generated by the AES peripheral, to signal the following events:

- computation completed
- read error, see [Section 32.4.17](#)
- write error, see [Section 32.4.17](#)

These three sources are combined into a common interrupt signal aes\_it that connects to NVIC (nested vectored interrupt controller).

**Figure 250. AES interrupt signal generation**



MSv42162V1

Each AES interrupt source can individually be enabled/disabled, by setting/clearing the corresponding enable bit of the AES\_CR register. See [Figure 250](#).

The status of the individual maskable interrupt sources can be read from the AES\_SR register.

[Table 205](#) gives a summary of the interrupt sources, their event flags and enable bits.

**Table 205. AES interrupt requests**

AES interrupt event	Event flag	Enable bit
computation completed flag	CCF	CCFIE
read error flag	RDERR	ERRIE
write error flag	WRERR	ERRIE

### 32.6 AES processing latency

The tables below summarize the latency to process a 128-bit block for each mode of operation.

**Table 206. Processing latency (in clock cycle) for ECB, CBC and CTR**

Key size	Mode of operation	Algorithm	Input phase + FSM set	Computation phase	Output phase	Total
128-bit	Mode 1: Encryption	ECB, CBC, CTR	9	38	4	<b>51</b>
	Mode 2: Key derivation	-	-	59	-	<b>59</b>
	Mode 3: Decryption	ECB, CBC, CTR	9	38	4	<b>51</b>
	Mode 4: Key derivation then decryption	ECB, CBC	9	93	4	<b>106</b>
256-bit	Mode 1: Encryption	ECB, CBC, CTR	13	58	4	<b>75</b>
	Mode 2: Key derivation	-	-	82	-	<b>82</b>
	Mode 3: Decryption	ECB, CBC, CTR	13	58	4	<b>75</b>
	Mode 4: Key derivation then decryption	ECB, CBC	13	128	4	<b>145</b>

**Table 207. Processing latency for GCM and CCM (in clock cycle)**

Key size	Mode of operation	Algorithm	Init Phase	Header phase	Payload phase	Tag phase
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

*Note: Data insertion can include wait states forced by AES on the AHB bus (maximum 3 cycles, typical 1 cycle). This applies to all header/payload/tag phases*



## 32.7 AES registers

### 32.7.1 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.	KEYSI ZE	Res.	CHMO D[2]	
								rw				rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GCM PH[1:0]		DMAO UTEN	DMAIN EN	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

Bit 31:24 Reserved, must be kept at zero

Bit 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 zero.

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

The bit value change is allowed only when AES is disabled, so as to avoid an unpredictable behavior.

Bit 17 Reserved, must be kept at zero

Bit 16 **CHMOD[2]**: Chaining mode selection, bit [2]

Refer to the bits [5:6] of the register for the description of the CHMOD[2:0] bitfield

Bit 15 Reserved, must be kept at zero

Bits 14:13 **GCM PH[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).

Usage 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).

Usage 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 6:5 **CHMOD[1:0]**: Chaining mode selection, bits [1:0]

These bits, together with the bit CHMOD[2] (see bit 16 of this register), form CHMOD[2:0] bitfield that 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)
- >100: Reserved

The bitfield value change is allowed only when AES is disabled, so as to avoid an unpredictable behavior.

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

The bitfield value change is allowed only when AES is disabled, so as to avoid an unpredictable behavior. 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 32.4.13: AES data registers and data swapping](#).

The bitfield value change is allowed only when AES is disabled, so as to avoid an unpredictable behavior.

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 when the key preparation process ends (Mode 2).

### 32.7.2 AES status register (AES\_SR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bits 31:4 Reserved, must be kept at zero

Bit 3 **BUSY**: Busy

This flag indicates whether AES is idle or busy during GCM payload **encryption** phase:

0: Idle

1: Busy

The flag is controlled by hardware. When the flag indicates “idle”, the current message processing may be suspended to process a higher-priority message.

This flag is effective only in GCM payload encryption phase. In other chaining modes, or in GCM phases other than payload encryption, the flag must be ignored.

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.

The flag is not effective when key derivation mode, or GCM/CCM Init phase is selected.

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.

The flag is not effective when key derivation mode, nor GCM/CCM init/header phase is selected.

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.

### 32.7.3 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[x+31:x+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[x+15:x]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DIN[x+31:x]**: One of four 32-bit words of a 128-bit input data block being written into the peripheral

This bitfield feeds a 32-bit input buffer. A 4-fold sequential write to this bitfield during the input phase virtually writes a complete 128-bit block of input data to the AES peripheral. Upon each write, the data from the 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 substitution for “x”, from the first to the fourth write operation, is: 96, 64, 32, and 0. In other words, data from the first to the fourth write operation are: DIN[127:96], DIN[95:64], DIN[63:32], and DIN[31:0].

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 32.4.13: AES data registers and data swapping on page 1039](#).



### 32.7.4 AES data output register (AES\_DOUTR)

Address offset: 0x0C

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
DOUT[x+31:x+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[x+15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DOUT[x+31:x]**: One of four 32-bit words of a 128-bit output data block being read from the peripheral

This read-only bitfield fetches a 32-bit output buffer. A 4-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.

The substitution for DOUT[x+31:x], from the first to the fourth read operation, is: 96, 64, 32, and 0. In other words, data from the first to the fourth read operation are: DOUT[127:96], DOUT[95:64], DOUT[63:32], and DOUT[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 32.4.13: AES data registers and data swapping on page 1039](#).

### 32.7.5 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]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **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 the AES peripheral is disabled.

Refer to [Section 32.4.14: AES key registers on page 1041](#) for more details.

### 32.7.6 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]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[47:32]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]

Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.

### 32.7.7 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]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[79:64]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]

Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.

### 32.7.8 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]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[111:96]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]  
 Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.

### 32.7.9 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]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]  
 Refer to [Section 32.4.15: AES initialization vector registers on page 1041](#) for description of the IVI[127:0] bitfield.  
 The initialization vector is only used in chaining modes other than ECB.  
 The initialization vector may be written only when the AES peripheral is disabled.

### 32.7.10 AES initialization vector register 1 (AES\_IVR1)

Address offset: 0x24  
 Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **IVI[63:32]**: Initialization vector input, bits [63:32]  
 Refer to the AES\_IVR0 register for description of the IVI[128:0] bitfield.

### 32.7.11 AES initialization vector register 2 (AES\_IVR2)

Address offset: 0x28



Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **IVI[95:64]**: Initialization vector input, bits [95:64]  
 Refer to the AES\_IVR0 register for description of the IVI[128:0] bitfield.

### 32.7.12 AES initialization vector register 3 (AES\_IVR3)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **IVI[127:96]**: Initialization vector input, bits [127:96]  
 Refer to the AES\_IVR0 register for description of the IVI[128:0] bitfield.

### 32.7.13 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]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[143:128]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]  
 Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.

### 32.7.14 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.

### 32.7.15 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.

### 32.7.16 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).

### 32.7.17 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
SUSPx															
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
SUSPx															
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Bits 31:0 **SUSPx**: AES suspend

Upon suspend operation, this bitfield of every AES\_SUSPxR register takes the value of one of internal AES registers.

32.7.18 AES register map

Table 208. AES register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
0x0000	AES_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPBLB	Res.	Res.	KEYSIZE	Res.	Res.	CHMOD[2]	Res.	GCMPIH[1:0]	Res.	DMAOUTEN	DMAINEN	ERRIE	CCFIE	ERRC	CCFC	Res.	CHMOD[1:0]	Res.	MODE[1:0]	Res.	DATATYPE[1:0]	Res.	EN		
	Reset value										0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0004	AES_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value																													0	BUSY	0	WRERR	0	RDERR	0
0x0008	AES_DINR x=96,64,32,0	DIN[x+31:x]																																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x000C	AES_DOUTR x=96,64,32,0	DOUT[x+31:x]																																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0010	AES_KEYR0	KEY[31:0]																																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0014	AES_KEYR1	KEY[63:32]																																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0018	AES_KEYR2	KEY[95:64]																																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x001C	AES_KEYR3	KEY[127:96]																																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0020	AES_IVR0	IVI[31:0]																																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0024	AES_IVR1	IVI[63:32]																																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0028	AES_IVR2	IVI[95:64]																																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x002C	AES_IVR3	IVI[127:96]																																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0030	AES_KEYR4	KEY[159:128]																																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0034	AES_KEYR5	KEY[191:160]																																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0038	AES_KEYR6	KEY[223:192]																																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x003C	AES_KEYR7	KEY[255:224]																																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0040	AES_SUSP0R	SUSP0[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	
0x0044	AES_SUSP1R	SUSP1[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	

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	
0x0048	AES_SUSP2R	SUSP2[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
0x004C	AES_SUSP3R	SUSP3[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
0x0050	AES_SUSP4R	SUSP4[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
0x0054	AES_SUSP5R	SUSP5[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
0x0058	AES_SUSP6R	SUSP6[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
0x005C	AES_SUSP7R	SUSP7[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



## 33 Hash processor (HASH)

### 33.1 Introduction

The hash processor is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA-224, SHA-256), the MD5 (message-digest algorithm 5) hash algorithm and the HMAC (keyed-hash message authentication code) algorithm suitable for a variety of applications. HMAC is suitable for applications requiring message authentication.

The hash processor computes FIPS (Federal Information Processing Standards) approved digests of length of 160, 224, 256 bits, for messages of up to  $(261 - 1)$  bits. It also computes 128 bits digests for the MD5 algorithm.

### 33.2 HASH main features

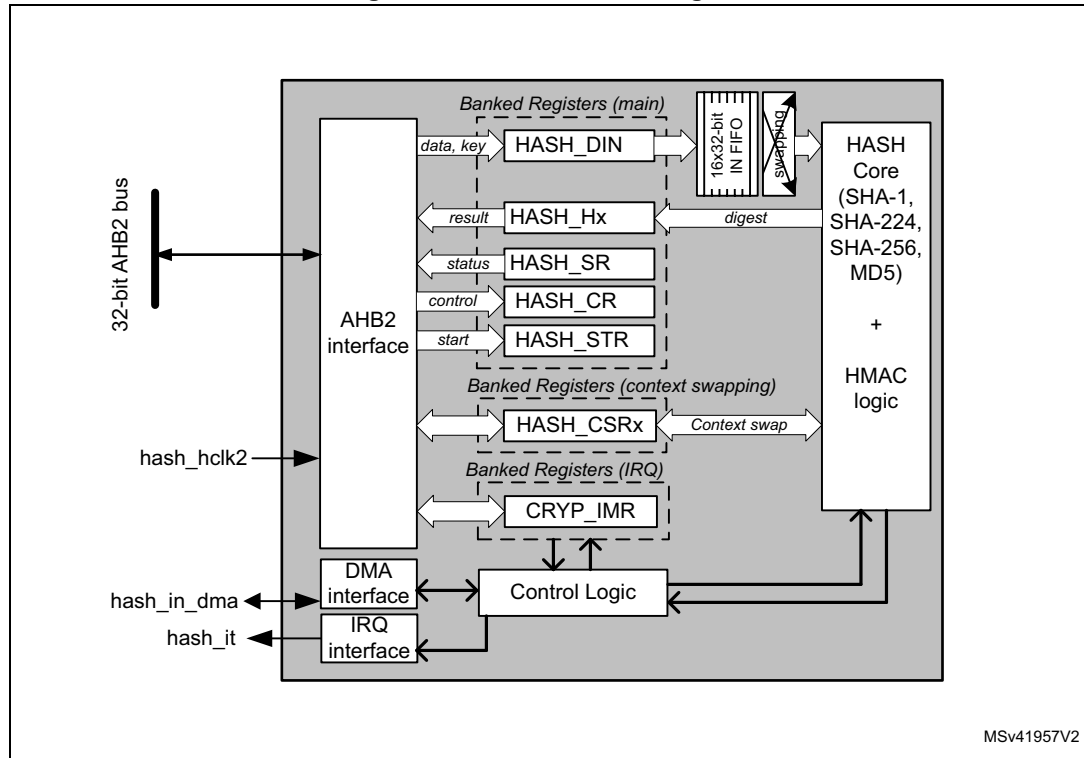
- Suitable for data authentication applications, compliant with:
  - Federal Information Processing Standards Publication FIPS PUB 180-4, *Secure Hash Standard* (SHA-1 and SHA-2 family)
  - Internet Engineering Task Force (IETF) Request For Comments RFC 1321 *MD5 Message-Digest Algorithm*
  - Internet Engineering Task Force (IETF) Request For Comments RFC 2104 *HMAC: Keyed-Hashing for Message Authentication*, and Federal Information Processing Standards Publication FIPS PUB 198-1, *The Keyed-Hash Message Authentication Code (HMAC)*
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
  - Automatic 32-bit words swapping to comply with the internal little-endian representation of the input bit-string
  - Word swapping supported: bits, bytes, half-words and 32-bit words
- Automatic padding to complete the input bit string to fit digest minimum block size of 512 bits ( $16 \times 32$  bits)
- Single 32-bit input register associated to an internal input FIFO of sixteen 32-bit words, corresponding to one block size
- Fast computation of SHA-1, SHA-224, SHA-256, and MD5
  - 82 (respectively 66) clock cycles for processing one 512-bit block of data using SHA-1 (respectively SHA-256) algorithm
  - 66 clock cycles for processing one 512-bit block of data using MD5 algorithm
- AHB slave peripheral, accessible through 32-bit word accesses only (else an AHB error is generated)
- $8 \times 32$ -bit words (H0 to H7) for output message digest
- Automatic data flow control with support of direct memory access (DMA) using one channel. Fixed burst of 4 supported.
- Interruptible message digest computation, on a per-32-bit word basis
  - Re-loadable digest registers
  - Hashing computation suspend/resume mechanism, including using DMA

### 33.3 HASH functional description

#### 33.3.1 HASH block diagram

Figure 251 shows the block diagram of the hash processor.

Figure 251. HASH block diagram



MSv41957V2

#### 33.3.2 HASH internal signals

Table 209 describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

Table 209. HASH internal input/output signals

Signal name	Signal type	Description
hash_hclk2	digital input	AHB2 bus clock
hash_it	digital output	Hash processor global interrupt request
hash_in_dma	digital input/output	DMA burst request/ acknowledge

### 33.3.3 About secure hash algorithms

The hash processor is a fully compliant implementation of the secure hash algorithm defined by FIPS PUB 180-4 standard and the IETF RFC1321 publication (MD5).

With each algorithm, the HASH computes a condensed representation of a message or data file. More specifically, when a message of any length below  $2^{64}$  bits is provided on input, the SHA-1, SHA-224, SHA-256 and MD5 processing core produces respectively a 160-bit, 224 bit, 256 bit and 128-bit output string called a message digest. The message digest can then be processed with a digital signature algorithm in order to generate or verify the signature for the message.

Signing the message digest rather than the message often improves the efficiency of the process because the message digest is usually much smaller in size than the message. The verifier of a digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-2 functions supported by the hash processor are qualified as “secure” because it is computationally infeasible to find a message that corresponds to a given message digest (SHA-1 is no more qualified as secure since February 2017), or to find two different messages that produce the same message digest. Any change to a message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify.

### 33.3.4 Message data feeding

The message (or data file) to be processed by the HASH should be considered as a bit string. Per FIPS PUB 180-1 and 180-2 standards this message bit string grows from left to right, with hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most bit position. For example message string “abc” with a bit string representation of “01100001 01100010 01100011” is represented by a 32-bit word `0x00636261`, and 8-bit words `0x61626300`.

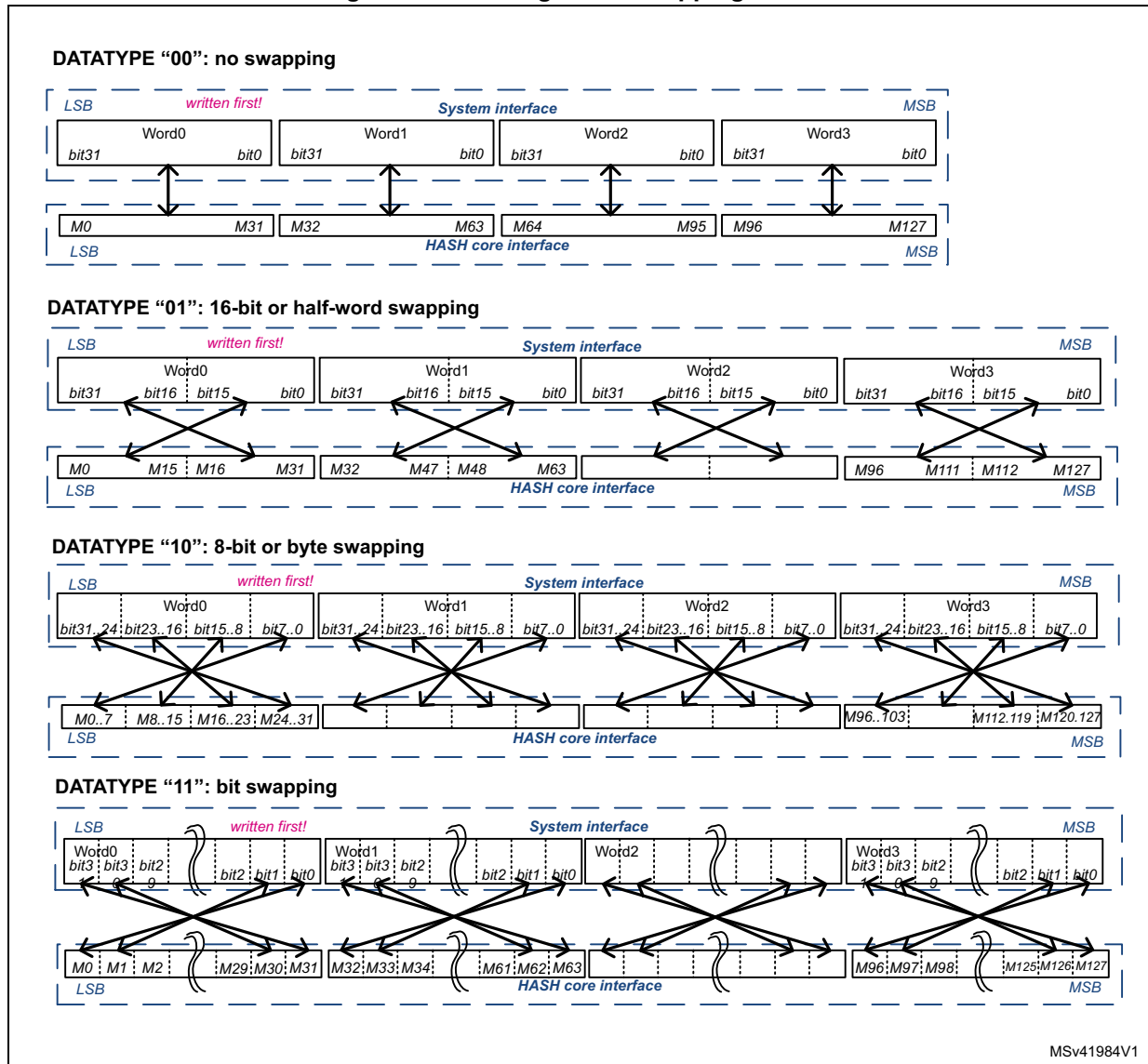
Data are entered into the HASH one 32-bit word at a time, by writing them into the HASH\_DIN register. The current contents of the HASH\_DIN register are transferred to the 16 words input FIFO (IN FIFO) each time the register is written with new data. Hence HASH\_DIN and the input FIFO form a seventeen 32-bit words length FIFO (named the IN buffer).

In accordance to the kind of data to be processed (e.g. byte swapping when data are ASCII text stream) there must be a bit, byte, half-word or no swapping operation to be performed on data from the input FIFO before entering the little-endian hash processing core.

[Figure 252](#) shows how the hash processing core 32-bit data block M0...31 is constructed from one 32-bit words popped into IN FIFO by the driver, according to the DATATYPE bitfield in the HASH control register (HASH\_CR).

HASH\_DIN data endianness when bit swapping is disabled (DATATYPE=“00”) can be described as following: the least significant bit of the message has to be at MSB position in the first word entered into the hash processor, the 32nd bit of the bit string has to be at MSB position in the second word entered into the hash processor and so on.

Figure 252. Message data swapping feature



### 33.3.5 Message digest computing

The hash processor sequentially processes 512-bit blocks when computing the message digest. Thus, each time  $16 \times 32$ -bit words (= 512 bits) have been written to the hash processor by the DMA or the CPU, the HASH automatically starts computing the message digest. This operation is known as 'partial digest computation'.

As described in [Section 33.3.4: Message data feeding](#), the message to be processed is entered into the HASH 32-bit word at a time, writing to the HASH\_DIN register to fill the input FIFO. In order to perform the hash computation on this data below sequence shall be used by the application.

1. Initialize the hash processor using the HASH\_CR register:
  - Select the right algorithm using ALGO field. If needed program the correct swapping operation on the message input words using DATATYPE bitfield in HASH\_CR.
  - Set MODE=1 and select the key length using LKEY if HMAC mode has been selected.
  - Update NBLW to define the number of valid bits in last word if it is different from 32 bits. If it is the case automatic padding could be applied by the HASH.
2. Complete the initialization by setting to 1 the INIT bit in HASH\_CR. Also set the bit DMAE to 1 if data are transferred via DMA.

**Caution:** When programming step 2, it is important to set up before or at the same time the correct configuration values (ALGO, DATATYPE, HMAC mode, key length, NBLW).

3. Start filling data by writing to HASH\_DIN register, unless data are automatically transferred via DMA. Note that the processing of a block can start only once the last value of the block has entered the IN FIFO. The way the partial or final digest computation is managed depends on the way data are fed into the processor:
  - a) When data are filled by software:
    - The partial digest computation is triggered when the software writes an additional word to the HASH\_DIN register (actually the first word of the next block). Once the processor is ready again (DINIS=1 in HASH\_SR), the software can write new data to HASH\_DIN. This mechanism avoids the introduction of wait states by the HASH.
    - The final digest computation is triggered when the last block is entered and the software writes the DCAL bit to 1. If the message length is not an exact multiple of 512 bits, the NBLW field in HASH\_STR register must be written prior to writing DCAL bit (see [Section 33.3.6](#) for details).
  - b) When data are filled by DMA as a single DMA transfer (MDMAT bit="0"):
    - The partial digest computation is triggered automatically each time the FIFO is full.
    - The final digest computation is triggered automatically when the last block has been transferred to the HASH\_DIN register (DCAL bit is set to 1 by hardware). If the message length is not an exact multiple of 512 bits, the NBLW field in HASH\_STR register must be written prior to enabling the DMA (see [Section 33.3.6](#) for details).
  - c) When data are filled using multiple DMA transfers (MDMAT bit="1") :
    - The partial digest computations are triggered as for single DMA transfers. However the final digest computation is not triggered automatically when the last block has been transferred to the HASH\_DIN register (DCAL bit is not set to 1 by hardware). It allows the hash processor to receive a new DMA transfer as part of

this digest computation. To launch the final digest computation, the software must set MDMAT bit to 0 before the last DMA transfer in order to trigger the final digest computation as it is done for single DMA transfers (see description before).

4. Once computed, the digest can be read from the output registers as described in [Table 210](#).

**Table 210. Hash processor outputs**

Algorithm	Valid output registers	Most significant bit	Digest size (in bits)
MD5	HASH_H0 to HASH_H3	HASH_H0[31]	128
SHA-1	HASH_H0 to HASH_H4	HASH_H0[31]	160
SHA-224	HASH_H0 to HASH_H6	HASH_H0[31]	224
SHA-256	HASH_H0 to HASH_H7	HASH_H0[31]	256

For more information about HMAC detailed instructions, refer to [Section 33.3.7: HMAC operation](#).

### 33.3.6 Message padding

#### Overview

When computing a condensed representation of a message, the process of feeding data into the hash processor (with automatic partial digest computation every 512-bit block) loops until the last bits of the original message are written to the HASH\_DIN register.

As the length (number of bits) of a message can be any integer value, the last word written to the hash processor may have a valid number of bits between 1 and 32. This number of valid bits in the last word, NBLW, has to be written to the HASH\_STR register, so that message padding is correctly performed before the final message digest computation.

#### Padding processing

Detailed padding sequences with DMA is enabled or disabled are described in [Section 33.3.5: Message digest computing](#).

#### Padding example

As specified by Federal Information Processing Standards PUB 180-1 and PUB 180-2, message padding consists in appending a “1” followed by  $k$  “0”s, itself followed by a 64-bit integer that is equal to the length  $L$  in bits of the message. These three padding operations generate a padded message of length  $L + 1 + k + 64$ , which by construction is a multiple of 512 bits.

For the hash processor, the “1” is added to the last word written to the HASH\_DIN register at the bit position defined by the NBLW bitfield, and the remaining upper bits are cleared (“0”s).

**Example from FIPS PUB180-2**

Let us assume that the original message is the ASCII binary-coded form of “abc”, of length  $L = 24$ :

```

byte 0   byte 1   byte 2   byte 3
01100001 01100010 01100011 UUUUUUUU
<-- 1st word written to HASH_DIN -->

```

NBLW has to be loaded with the value 24: a “1” is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH\_DIN register (little-endian convention):

```
01100001 01100010 01100011 1UUUUUUU
```

Since  $L = 24$ , the number of bits in the above bit string is 25, and 423 “0” bits are appended, making now 448 bits.

This gives in hexadecimal (byte words in big-endian format):

```

61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000

```

The message length value,  $L$ , in two-word format (that is 00000000 00000018) is appended. Hence the final padded message in hexadecimal (byte words in big-endian format):

```

61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018

```

If the hash processor is programmed to swap byte within HASH\_DIN input register (DATATYPE=10 in HASH\_CR), the above message has to be entered by following below the sequence:

1. 0xUU636261 is written to the HASH\_DIN register (where ‘U’ means don’t care).
2. 0x18 is written to the HASH\_STR register (the number of valid bits in the last word written to the HASH\_DIN register is 24, as the original message length is 24 bits).
3. 0x10 is written to the HASH\_STR register to start the message padding (described above) and then perform the digest computation.
4. The hash computing is complete with the message digest available in the HASH\_HRx registers ( $x = 0\dots4$ ) for the SHA-1 algorithm. For this FIPS example, the expected value is as follows:

```

HASH_H0 = 0xA9993E36
HASH_H1 = 0x4706816A
HASH_H2 = 0xBA3E2571
HASH_H3 = 0x7850C26C
HASH_H4 = 0x9CD0D89D

```

**33.3.7 HMAC operation****Overview**

As specified by Internet Engineering Task Force *RFC2104, HMAC: keyed-hashing for message authentication*, the HMAC algorithm is used for message authentication by irreversibly binding the message being processed to a key chosen by the user. The algorithm consists of two nested hash operations:

$$\text{HMAC}(\text{message}) = \text{Hash}((\text{key} \mid \text{pad}) \text{ XOR } [0x5C]_n \mid \text{Hash}((\text{key} \mid \text{pad}) \text{ XOR } [0x36]_n \mid \text{message}))$$

where:

- $[X]_n$  represents a repetition of X n times, where n equal to the size of the underlying hash function data block that is 512 bits for SHA-1, SHA224, SHA-256, MD5 hash algorithms (i.e. n=64).
- pad is a sequence of zeroes needed to extend the key to the length n defined above. If the key length is greater than n, the application shall first hash the key using Hash() function and then use the resultant byte string as the actual key to HMAC.
- | represents the concatenation operator.

### HMAC processing

Four different steps are required to compute the HMAC:

1. The block is initialized by writing the INIT bit to 1 with the MODE bit at 1 and the ALGO bits set to the value corresponding to the desired algorithm. The LKEY bit must also be set to 1 if the key being used is longer than 64 bytes. In this case, as required by HMAC specifications, the hash processor will use the hash of the key instead of the real key.
2. The key to be used for the inner hash function must be provided to the hash processor: The key loading operation follows the same mechanism as the message bit string loading, i.e. write key data into HASH\_DIN and complete the transfer by writing to HASH\_STR register.

*Note:* Endianness details can be found in [Section 33.3.4: Message data feeding](#).

3. Once the last key word has been entered and computation has started, the hash processor elaborates the inner key material. Once this operation has completed, it is ready to accept the message bit string as described in [Section 33.3.4: Message data feeding](#).
4. After the final hash round, the hash processor returns “ready” to indicate that it is ready to receive the key to be used for the outer hash function (normally, this key is the same as the one used for the inner hash function). When the last word of the key is entered and computation starts, the HMAC result can be found in the HASH\_H0...HASH\_H7 registers.

*Note:* The computation latency of the HMAC primitive depends on the lengths of the keys and message, as described in [Section 33.5: HASH processing time](#).

### HMAC example

Below is an example of HMAC SHA-1 algorithm (ALGO="00" and MODE="1" in HASH\_CR) as specified by NIST.

Let us assume that the original message is the ASCII binary-coded form of “sample message for keylen=blocklen”, of length L = 34 bytes. If the HASH is programmed in no swapping mode (DATATYPE=00 in HASH\_CR), the following data must be loaded sequentially into HASH\_DIN register:

1. **Inner hash key** input (length=64, i.e. no padding), specified by NIST. As key length=64, LKEY bit is set to 0 in HASH\_CR register

```
00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617
18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F
```



30313233 34353637 38393A3B 3C3D3E3F

2. **Message** input (length=34, i.e. padding required). HASH\_STR must be set to 0x20 to start message padding and inner hash computation (see 'U' as don't care)

53616D70 6C65206D 65737361 67652066 6F72206B 65796C65  
6E3D626C 6F636B6C 656E0000

3. **Outer hash key** input (length=64, i.e. no padding). A key identical to the inner hash key is entered here.

4. **Final outer hash computing** is then performed by the HASH. The HMAC SHA-1 digest result is available in the HASH\_HRx registers (x = 0...4), as shown below:

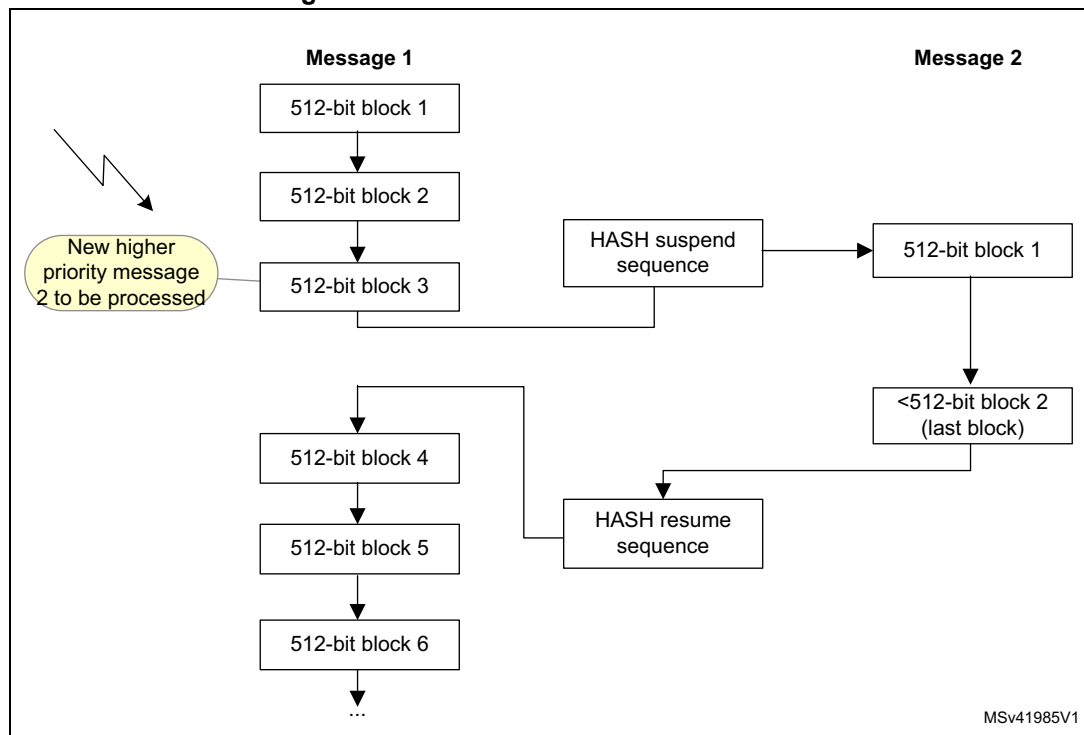
HASH\_H0 = 0x5FD596EE  
HASH\_H1 = 0x78D5553C  
HASH\_H2 = 0x8FF4E72D  
HASH\_H3 = 0x266DFD19  
HASH\_H4 = 0x2366DA29

### 33.3.8 Context swapping

#### Overview

It is possible to interrupt a hash/HMAC operation to perform another processing with a higher priority. The interrupted process completes later when the higher-priority task has been processed, as shown in [Figure 253](#).

Figure 253. HASH save/restore mechanism



To do so, the context of the interrupted task must be saved from the HASH registers to memory, and then be restored from memory to the HASH registers.

The procedures where the data flow is controlled by software or by DMA are described below.

### Data loaded by software

When the DMA is not used to load the message into the hash processor, the context can be saved only when no block processing is ongoing. This means that the user application must wait until  $DINIS = 1$  (last block processed and input FIFO empty) or  $NBW \neq 0$  (FIFO not full and no processing ongoing). The detailed procedure is described below.

- **Current context saving**

Before interrupting the current message digest calculation, the application must store the contents of the following registers into memory:

- HASH\_IMR
- HASH\_STR
- HASH\_CR
- HASH\_CSR0 to HASH\_CSR53

- **Current context restoring**

To resume processing the interrupted message, the application must respect the following steps:

- a) Write the following registers with the values saved in memory: HASH\_IMR, HASH\_STR and HASH\_CR.
- b) Initialize the hash processor by setting the INIT bit in the HASH\_CR register.
- c) Write the HASH\_CSR0 to HASH\_CSR53 registers with the values saved in memory.
- d) Restart the processing from the point where it has been interrupted.

### Data loaded by DMA

When the DMA is used to load the message into the hash processor, it is not possible to predict if a DMA transfer is ongoing. The user application must thus stop DMA transfers, then wait until the hash processor is ready before interrupting the current message digest calculation. The detailed procedure is described below.

- **Current context saving**

Before interrupting the current message digest calculation using DMA, the application must respect the following steps:

- a) Clear the DMAE bit to disable the DMA interface.
- b) Wait until the current DMA transfer is complete (wait for  $DMAS = 0$  in the HASH\_SR register). Note that the block may or may not have been totally transferred to the HASH.
- c) Disable the corresponding channel in the DMA controller.
- d) Wait until the hash processor is ready (no block is being processed), that is wait for  $DINIS = 1$

- **Current context restoring**

To resume processing the interrupted message using DMA, the application must respect the following steps:

- a) Reconfigure the DMA controller so that it proceeds with the transfer of the message up to the end if it is not interrupted again.
- b) Restart the processing from the point where it was interrupted by setting the DMAE bit.

*Note:* If the context swapping does not involve HMAC operations, the HASH\_CSR38 to HASH\_CSR53 registers do not need to be saved and restored.

If the context swapping occurs between two blocks (the last block was completely processed and the next block has not yet been pushed into the IN FIFO, NBW = 000 in the HASH\_CR register), the HASH\_CSR22 to HASH\_CSR37 registers do not need to be saved and restored.

### 33.3.9 HASH DMA interface

The hash processor provides an interface to connect to the DMA controller. This DMA can be used to write data to the HASH by setting the DMAE bit in the HASH\_CR register. When this bit is set, the HASH asserts the burst request signal to the DMA controller when there is enough free words in the FIFO to support a burst of four words.

Once four 32-bit words have been received, the HASH automatically restarts this process, checks the FIFO size, and asserts a new request if the FIFO status allow a burst reception. For more information refer to [Section 33.3.5: Message digest computing](#).

Before starting the DMA transfer, the software must program the number of valid bits in the last word that will be copied into HASH\_DIN register. This is done by writing in HASH\_STR register the following value:

$$NBLW = \text{Len}(\text{Message}) \% 32$$

where “x%32” gives the remainder of x divided by 32.

DMAS bit in HASH\_SR register provides information on the DMA interface activity. This bit is set with DMAE and cleared when DMAE is cleared to 0 and no DMA transfer is ongoing.

*Note:* No interrupt is associated to DMAS bit.

### 33.3.10 HASH error management

No error flags are generated by the HASH hardware.

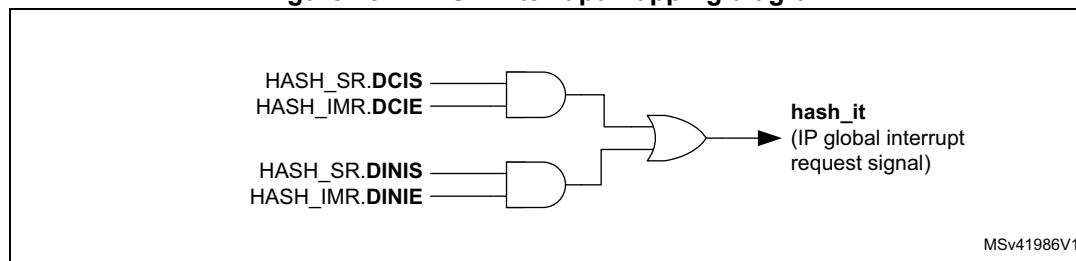
## 33.4 HASH interrupts

Two individual maskable interrupt sources are generated by the hash processor to signal following events:

- Digest calculation completion (DCIS)
- Data input buffer ready (DINIS)

Both interrupt sources are connected to the same global interrupt request signal, as shown on [Figure 254](#).

**Figure 254. HASH interrupt mapping diagram**



The above interrupt sources can be enabled or disabled individually by changing the mask bits in the HASH\_IMR register. Setting the appropriate mask bit to 1 enables the interrupt.

The status of the individual interrupt events can be read from the HASH\_SR register.

[Table 211](#) gives a summary of the available features.

**Table 211. HASH interrupt requests**

Interrupt event	Event flag	Enable control bit
Digest computation completed flag	DCIS	DCIE
Data input buffer ready to get a new block flag	DINIS	DINIE

### 33.5 HASH processing time

[Table 212](#) summarizes the time required to process a 512-bit intermediate block for each mode of operation.

**Table 212. Processing time (in clock cycle)**

Mode of operation	FIFO load <sup>(1)</sup>	Computation phase	Total
MD5	16	50	<b>66</b>
SHA-1	16	66	<b>82</b>
SHA-224	16	50	<b>66</b>
SHA-256			

1. The time required to load the 16 words of the block into the processor must be added to this value.

The time required to process the last block of a message (or of a key in HMAC) can be longer. This time depends on the length of the last block and the size of the key (in HMAC mode).

Compared to the processing of an intermediate block, it can be increased by the factor below:

- **1 to 2.5** for a hash message
- **~2.5** for an HMAC input-key
- **1 to 2.5** for an HMAC message
- **~2.5** for an HMAC output key in case of a short key
- **3.5 to 5** for an HMAC output key in case of a long key

### 33.6 HASH registers

The HASH core is associated with several control and status registers and five message digest registers. All these registers are accessible through 32-bit word accesses only, else an AHB2 error is generated.

#### 33.6.1 HASH control register (HASH\_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[1]	Res.	LKEY
													rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MDMAT	DINNE	NBW[3:0]				ALGO[0]	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
		rw	r	r	r	r	r	rw	rw	rw	rw	rw	w		

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

Bit 17 Reserved, must be kept at reset value.

Bit 16 **LKEY**: Long key selection

This bit selects between short key ( $\leq 64$  bytes) or long key ( $> 64$  bytes) in HMAC mode.

0: Short key ( $\leq 64$  bytes)

1: Long key ( $> 64$  bytes)

*Note: This selection is only taken into account when the INIT bit is set and MODE= 1. Changing this bit during a computation has no effect.*

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **MDMAT**: Multiple DMA Transfers

This bit is set when hashing large files when multiple DMA transfers are needed.

0: DCAL is automatically set at the end of a DMA transfer.

1: DCAL is not automatically set at the end of a DMA transfer.

Bit 12 **DINNE**: DIN not empty

This bit is set when the HASH\_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is written to 1.

0: No data are present in the data input buffer

1: The input buffer contains at least one word of data

Bits 11:8 **NBW[3:0]**: Number of words already pushed

This bitfield reflects the number of words in the message that have already been pushed into the IN FIFO. NBW increments (+1) when a write access is performed to the HASH\_DIN register while DINNE = 1.

It goes to zero when the INIT bit is written to 1 or when a digest calculation starts (DCAL written to 1 or DMA end of transfer).

**If the DMA is not used**

0000 and DINNE=0: no word has been pushed into the DIN buffer, i.e. both HASH\_DIN register and IN FIFO are empty.

0000 and DINNE=1: one word has been pushed into the DIN buffer, i.e. HASH\_DIN register contains one word and IN FIFO is empty.

0001: two words have been pushed into the DIN buffer, i.e. HASH\_DIN register and the IN FIFO contain one word each.

...

1111: 16 words have been pushed into the DIN buffer.

**If the DMA is used**

NBW is the exact number of words that have been pushed into the IN FIFO by the DMA.

Bits 18, 7 **ALGO[1:0]**: Algorithm selection

These bits selects the SHA-1, SHA-224, SHA-256 or the MD5 algorithm:

00: SHA-1 algorithm selected

01: MD5 algorithm selected

10: SHA-224 algorithm selected

11: SHA-256 algorithm selected

*Note: This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.*

Bit 6 **MODE**: Mode selection

This bit selects the HASH or HMAC mode for the selected algorithm:

0: Hash mode selected

1: HMAC mode selected. LKEY must be set if the key being used is longer than 64 bytes.

*Note: This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.*

Bits 5:4 **DATATYPE[1:0]**: Data type selection

These bits define the format of the data entered into the HASH\_DIN register:  
00: 32-bit data. The data written to HASH\_DIN are directly used by the hash processing, without reordering.

01: 16-bit data or half-word. The data written to HASH\_DIN are considered as two half-words, and are swapped before being used by the hash processing.

10: 8-bit data or bytes. The data written to HASH\_DIN are considered as four bytes, and are swapped before being used by the hash processing.

11: bit data or bit-string. The data written to HASH\_DIN are considered as 32 bits (1st bit of the string at position 0), and are swapped before being used by the hash processing (first bit of the string at position 31).

**Bit 3 DMAE:** DMA enable

0: DMA transfers disabled

1: DMA transfers enabled. A DMA request is sent as soon as the hash core is ready to receive data.

After this bit is set it is cleared by hardware while the last data of the message is written to the hash processor.

Setting this bit to 0 while a DMA transfer is on-going is not aborting this current transfer. Instead, the DMA interface of the HASH remains internally enabled until the transfer is complete or INIT is written to 1.

Setting INIT bit to 1 does not clear DMAE bit.

**Bit 2 INIT:** Initialize message digest calculation

Writing this bit to 1 resets the hash processor core, so that the HASH is ready to compute the message digest of a new message.

Writing this bit to 0 has no effect. Reading this bit always return 0.

Bits 1:0 Reserved, must be kept at reset value.

### 33.6.2 HASH data input register (HASH\_DIN)

Address offset: 0x04

Reset value: 0x0000 0000

HASH\_DIN is the data input register. It is 32-bit wide. This register is used to enter the message by blocks of 512 bits. When the HASH\_DIN register is programmed, the value presented on the AHB databus is ‘pushed’ into the hash core and the register takes the new value presented on the AHB databus. To get a correct message format, the DATATYPE bits must have been previously configured in the HASH\_CR register.

When a block of 16 words has been written to the HASH\_DIN register, an intermediate digest calculation is launched:

- by writing new data into the HASH\_DIN register (the first word of the next block) if the DMA is not used (intermediate digest calculation),
- automatically if the DMA is used.

When the last block has been written to the HASH\_DIN register, the final digest calculation (including padding) is launched:

- by writing the DCAL bit to 1 in the HASH\_STR register (final digest calculation),
- automatically if the DMA is used and MDMAT bit is set to 0.

When a digest calculation (intermediate or final) is ongoing and a new write access to the HASH\_DIN register is performed, wait-states are inserted on the AHB2 bus until the hash calculation completes.

When the HASH\_DIN register is read, the last word written to this location is accessed (zero after reset).

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

Bits 31:0 **DATAIN[31:0]**: Data input

Reading this register returns the current register content.

Writing this register pushes the current register content into the IN FIFO, and the register takes the new value presented on the AHB databus.



### 33.6.3 HASH start register (HASH\_STR)

Address offset: 0x08

Reset value: 0x0000 0000

The HASH\_STR register has two functions:

- It is used to define the number of valid bits in the last word of the message entered in the hash processor (that is the number of valid least significant bits in the last data written to the HASH\_DIN register)
- It is used to start the processing of the last block in the message by writing the DCAL bit to 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
							w				r/w	r/w	r/w	r/w	r/w

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

Bit 8 **DCAL**: Digest calculation

Writing this bit to 1 starts the message padding, using the previously written value of NBLW, and starts the calculation of the final message digest with all data words written to the IN FIFO since the INIT bit was last written to 1. Reading this bit returns 0.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **NBLW[4:0]**: Number of valid bits in the last word

When the last word of the message bit string is written in HASH\_DIN register, the hash processor takes only the valid bits specified as below, after internal data swapping:

0x00: All 32 bits of the last data written are valid message bits i.e. M[31:0]

0x01: Only one bit of the last data written (after swapping) is valid i.e. M[0]

0x02: Only two bits of the last data written (after swapping) are valid i.e. M[1:0]

0x03: Only three bits of the last data written (after swapping) are valid i.e. M[2:0]

...

0x1F: Only 31 bits of the last data written (after swapping) are valid i.e. M[30:0]

The above mechanism is valid only if DCAL=0. If NBLW bits are written while DCAL is set to 1, the NBLW bitfield remains unchanged. In other words it is not possible to configure NBLW and set DCAL at the same time.

Reading NBLW bits returns the last value written to NBLW.

### 33.6.4 HASH digest registers (HASH\_HR0..7)

These registers contain the message digest result named as follows:

- HASH\_HR0, HASH\_HR1, HASH\_HR2, HASH\_HR3 and HASH\_HR4 registers return the SHA-1 digest result  
HASH\_HR5 to HASH\_HR7 registers are not used, and they are read as zero.
- HASH\_HR0, HASH\_HR1, HASH\_HR2 and HASH\_HR3 registers return A, B, C and D (respectively), as defined by MD5.  
HASH\_HR4 to HASH\_HR7 registers are not used, and they are read as zero.
- HASH\_HR0 to HASH\_HR6 registers return the SHA-224 digest result.  
HASH\_HR7 register is not used, and it is read as zero.
- HASH\_HR0 to HASH\_HR7 registers return the SHA-256 digest result.

In all cases, the digest most significant bit is stored in HASH\_HR0[31] and it is not used.

If a read access to one of these registers is performed while the hash core is calculating an intermediate digest or a final message digest (that is when the DCAL bit has been written to 1), then the read operation is stalled until the hash calculation completes.

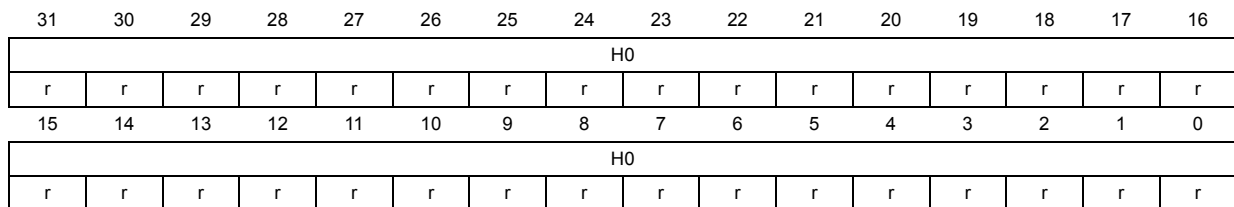
*Note:* HASH\_HR0, HASH\_HR1, HASH\_HR2, HASH\_HR3 and HASH\_HR4 mapping are duplicated in two memory regions.

#### HASH\_HR0

Address offset: 0x0C

Address offset: ALTERNATE: 0x310

Reset value: 0x0000 0000



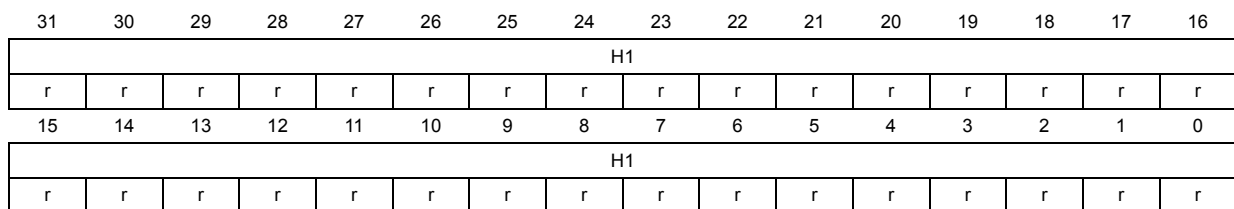
Bits 31:0 **H0**: refer to [Section 33.6.4: HASH digest registers \(HASH\\_HR0..7\)](#) introduction

#### HASH\_HR1

Address offset: 0x10

Address offset: ALTERNATE: 0x3104

Reset value: 0x0000 0000



Bits 31:0 **H1**: refer to [Section 33.6.4: HASH digest registers \(HASH\\_HR0..7\)](#) introduction

**HASH\_HR2**

Address offset: 0x14

Address offset: ALTERNATE: 0x3108

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H2															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H2															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **H2**: refer to [Section 33.6.4: HASH digest registers \(HASH\\_HR0..7\)](#) introduction

**HASH\_HR3**

Address offset: 0x18

Address offset: ALTERNATE: 0x31C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H3															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H3															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **H3**: refer to [Section 33.6.4: HASH digest registers \(HASH\\_HR0..7\)](#) introduction

**HASH\_HR4**

Address offset: 0x1C

Address offset: ALTERNATE: 0x320

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H4															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H4															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

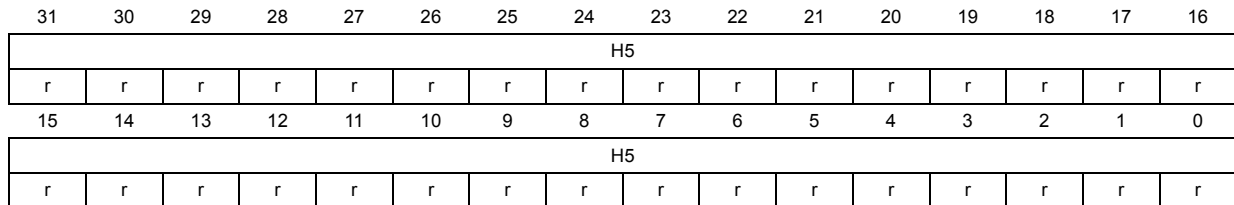


Bits 31:0 **H4**: refer to [Section 33.6.4: HASH digest registers \(HASH\\_HR0..7\)](#) introduction

**HASH\_HR5**

Address offset: 0x324

Reset value: 0x0000 0000

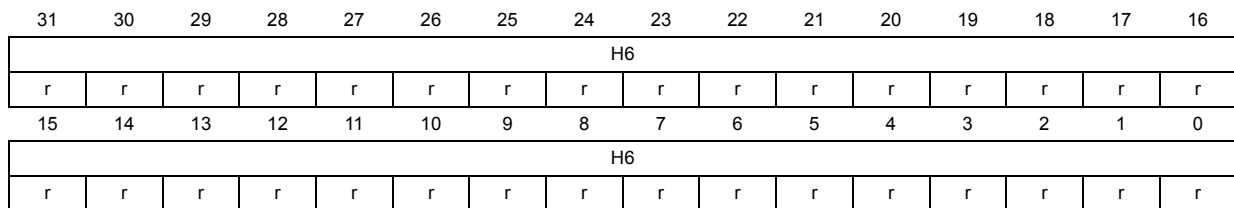


Bits 31:0 **H5**: refer to [Section 33.6.4: HASH digest registers \(HASH\\_HR0..7\)](#) introduction

**HASH\_HR6**

Address offset: 0x328

Reset value: 0x0000 0000

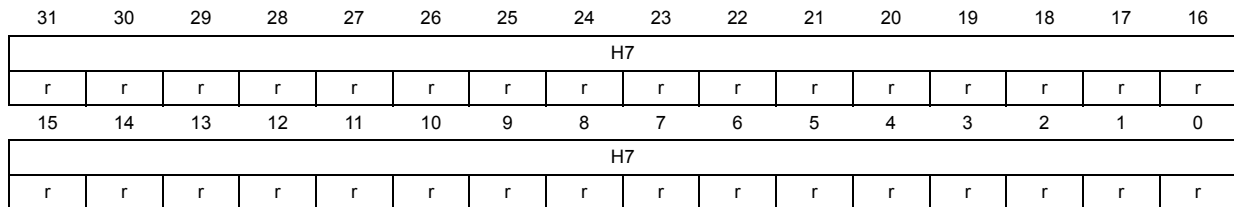


Bits 31:0 **H6**: refer to [Section 33.6.4: HASH digest registers \(HASH\\_HR0..7\)](#) introduction

**HASH\_HR7**

Address offset: 0x32C

Reset value: 0x0000 0000



Bits 31:0 **H7**: refer to [Section 33.6.4: HASH digest registers \(HASH\\_HR0..7\)](#) introduction

**Note:** When starting a digest computation for a new bit stream (by writing the INIT bit to 1), these registers are forced to their reset values.



### 33.6.5 HASH interrupt enable register (HASH\_IMR)

Address offset: 0x20

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DCIE**: Digest calculation completion interrupt enable

0: Digest calculation completion interrupt disabled

1: Digest calculation completion interrupt enabled.

Bit 0 **DINIE**: Data input interrupt enable

0: Data input interrupt disabled

1: Data input interrupt enabled

### 33.6.6 HASH status register (HASH\_SR)

Address offset: 0x24

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
												r	r	rc_w0	rc_w0

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

Bit 3 **BUSY**: Busy bit

- 0: No block is currently being processed
- 1: The hash core is processing a block of data

Bit 2 **DMAS**: DMA Status

This bit provides information on the DMA interface activity. It is set with DMAE and cleared when DMAE=0 and no DMA transfer is ongoing. No interrupt is associated with this bit.

- 0: DMA interface is disabled (DMAE=0) and no transfer is ongoing
- 1: DMA interface is enabled (DMAE=1) or a transfer is ongoing

Bit 1 **DCIS**: Digest calculation completion interrupt status

This bit is set by hardware when a digest becomes ready (the whole message has been processed). It is cleared by writing it to 0 or by writing the INIT bit to 1 in the HASH\_CR register.

- 0: No digest available in the HASH\_HRx registers
- 1: Digest calculation complete, a digest is available in the HASH\_HRx registers. An interrupt is generated if the DCIE bit is set in the HASH\_IMR register.

Bit 0 **DINIS**: Data input interrupt status

This bit is set by hardware when the input buffer is ready to get a new block (16 locations are free). It is cleared by writing it to 0 or by writing the HASH\_DIN register.

- 0: Less than 16 locations are free in the input buffer
- 1: A new block can be entered into the input buffer. An interrupt is generated if the DINIE bit is set in the HASH\_IMR register.

### 33.6.7 HASH context swap registers (HASH\_CSRx)

These registers contain the complete internal register states of the hash processor. They are useful when a context swap has to be done because a high-priority task needs to use the hash processor while it is already used by another task.

When such an event occurs, the HASH\_CSRx registers have to be read and the read values have to be saved in the system memory space. Then the hash processor can be used by the preemptive task, and when the hash computation is complete, the saved context can be read from memory and written back into the HASH\_CSRx registers.

#### HASH\_CSR0

Address offset: 0x0F8

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CS0															
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
CS0															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

#### HASH\_CSRx (x=1 to 53)

Address offset: 0x0F8 + x \* 0x4

Reset value: 0x0000 0000

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

### 33.6.8 HASH register map

Table 213 gives the summary HASH register map and reset values.

**Table 213. HASH 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	HASH_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[1]	Res.	LKEY	Res.	Res.	MDMAT	DINNE	NBW			Res.	ALGO[0]	MODE	DATATYPE	DMAE	INIT	Res.	Res.	
	Reset value														0		0			0	0	0	0	0	0	0	0	0	0	0	0	0	
0x04	HASH_DIN	DATAIN																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	HASH_STR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW				
	Reset value																								0				0	0	0	0	
0x0C	HASH_HR0	H0																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	HASH_HR1	H1																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	HASH_HR2	H2																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	HASH_HR3	H3																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	HASH_HR4	H4																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	HASH_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE	
	Reset value																													0	0		
0x24	HASH_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.	BUSY	DMAE	DCIS	DINIS	
	Reset value																												0	0	0	1	
0xF8	HASH_CSR0	CSR0																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0xFC	HASH_CSR1	CSR1																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x1CC	HASH_CSR53	CSR53																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
<b>Reserved</b>																																	
0x310	HASH_HR0	H0																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x314	HASH_HR1	H1																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x318	HASH_HR2	H2																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x31C	HASH_HR3	H3																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x320	HASH_HR4	H4																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x324	HASH_HR5	H5																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x328	HASH_HR6	H6																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x32C	HASH_HR7	H7																															
	Reset value	0	0	0	0	0	0	0	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 2.2.2 on page 82 for the register boundary addresses.





## 34 Advanced-control timers (TIM1/TIM8)

### 34.1 TIM1/TIM8 introduction

The advanced-control timers (TIM1/TIM8) consist of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

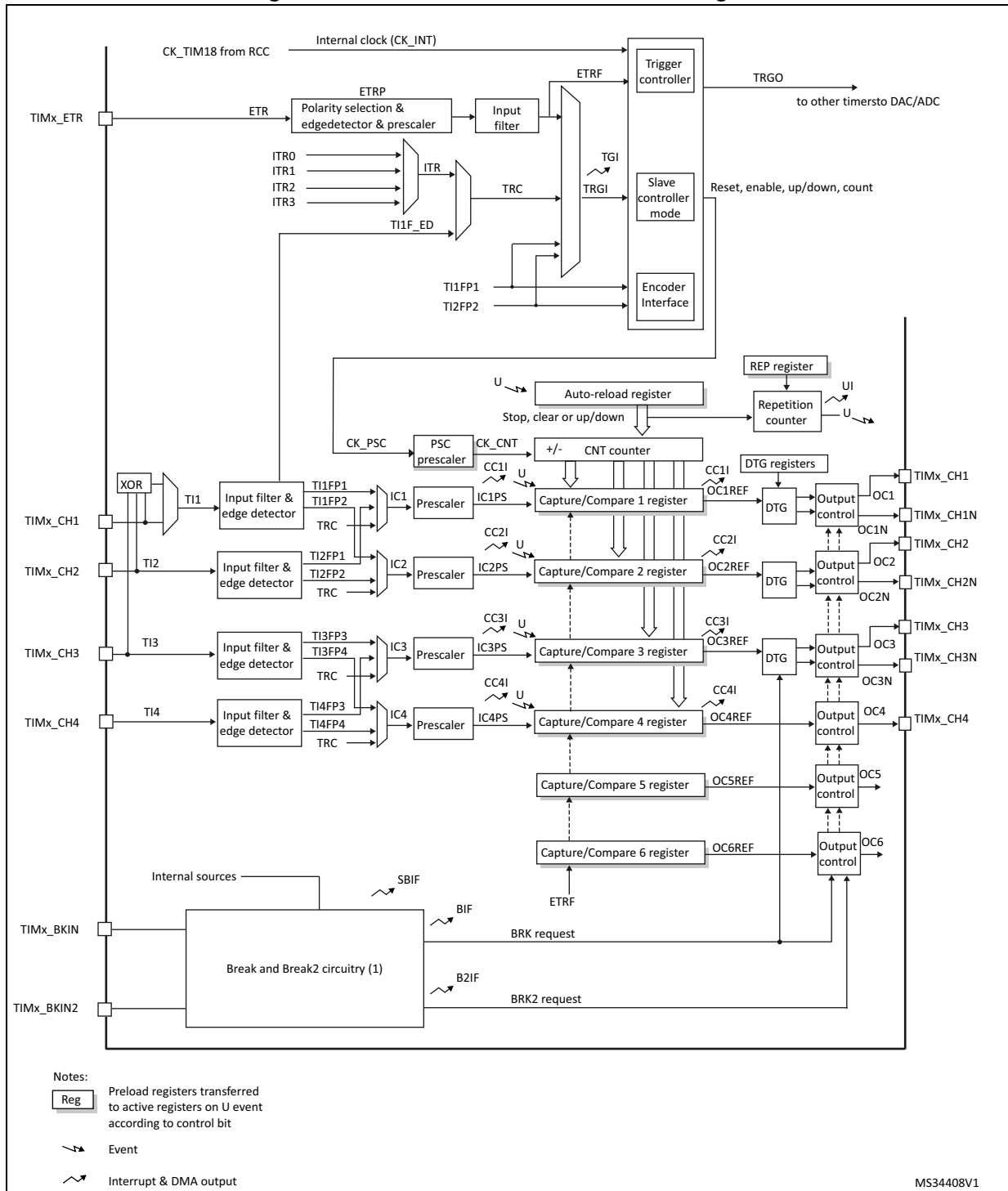
The advanced-control (TIM1/TIM8) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in [Section 34.3.26: Timer synchronization](#).

### 34.2 TIM1/TIM8 main features

TIM1/TIM8 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 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 255. Advanced-control timer block diagram



1. See [Figure 299: Break and Break2 circuitry overview](#) for details

## 34.3 TIM1/TIM8 functional description

### 34.3.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx\_CNT)
- Prescaler register (TIMx\_PSC)
- Auto-reload register (TIMx\_ARR)
- Repetition counter register (TIMx\_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx\_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx\_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIMx\_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx\_CR1 register.

#### Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx\_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 256* and *Figure 257* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 256. Counter timing diagram with prescaler division change from 1 to 2

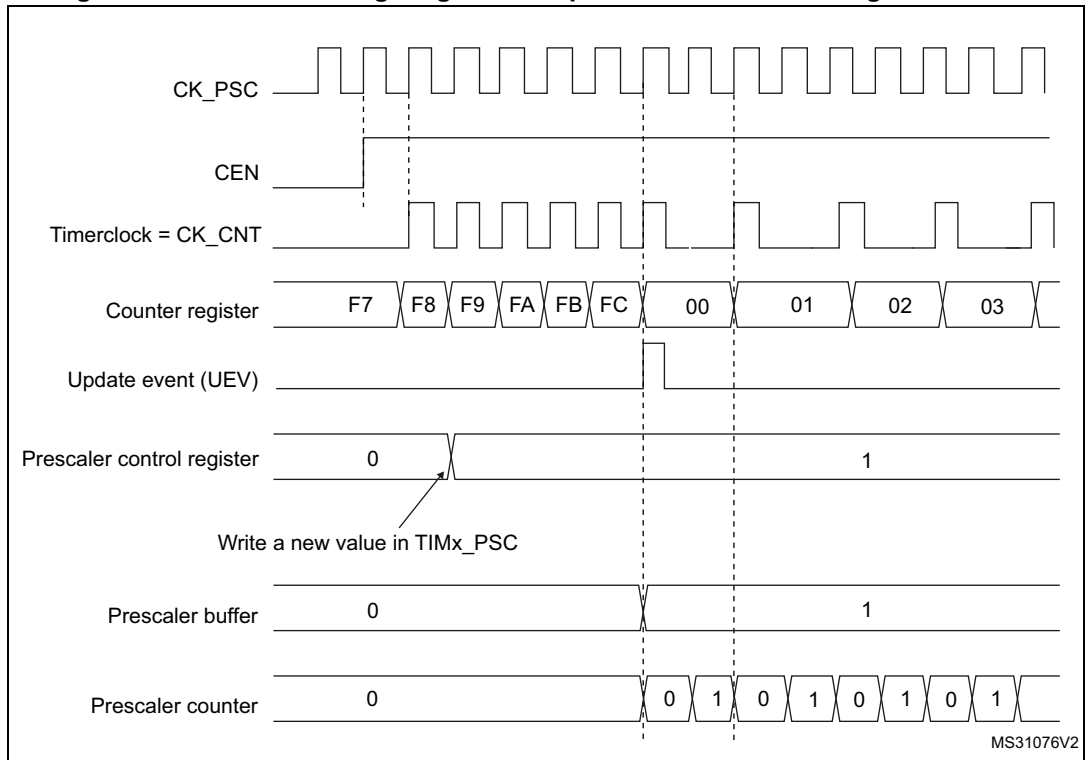
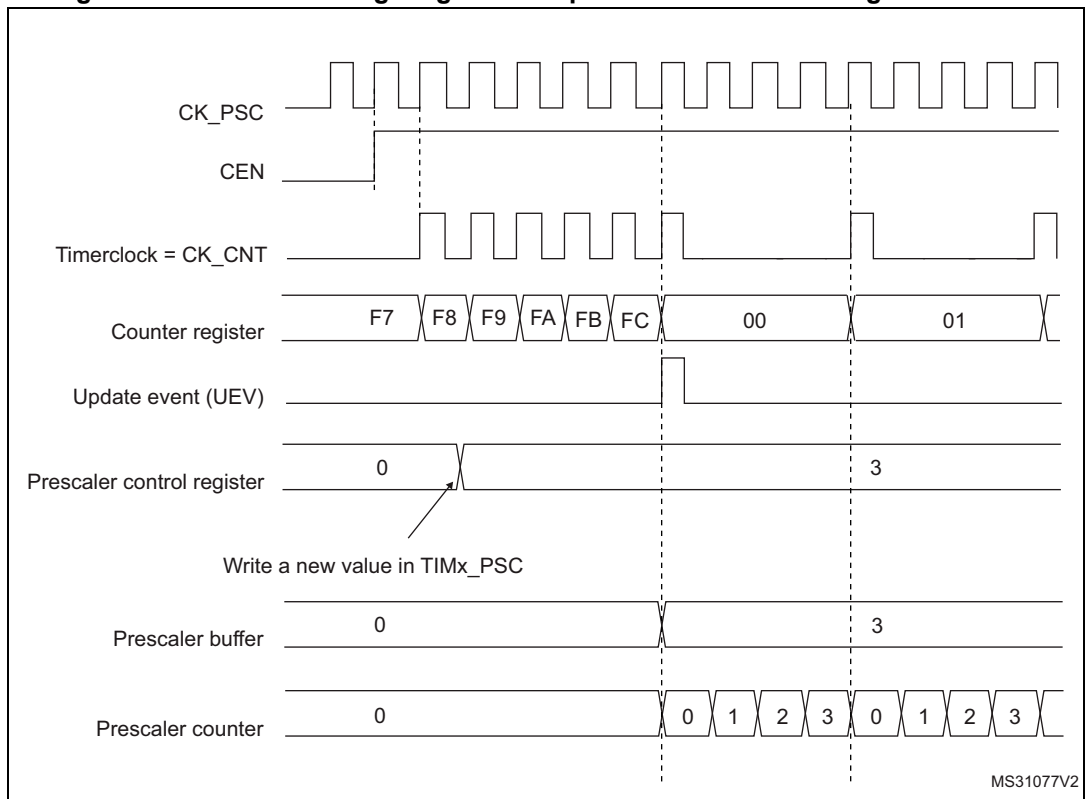


Figure 257. Counter timing diagram with prescaler division change from 1 to 4



### 34.3.2 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 258. Counter timing diagram, internal clock divided by 1

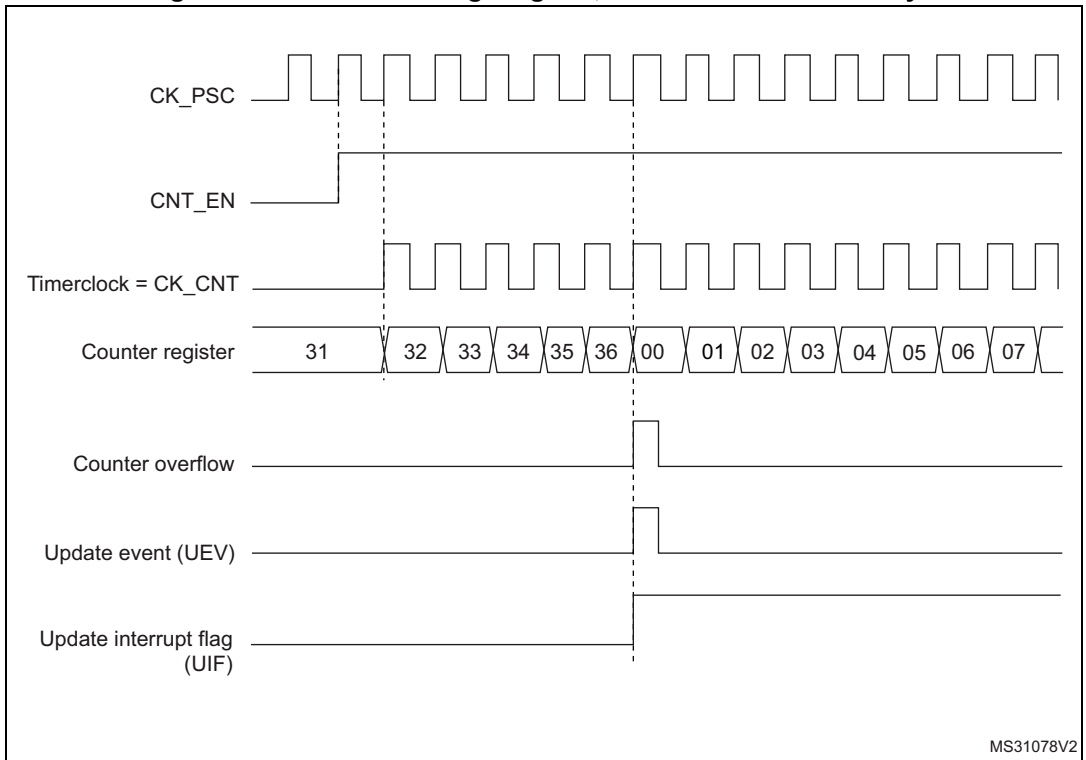


Figure 259. Counter timing diagram, internal clock divided by 2

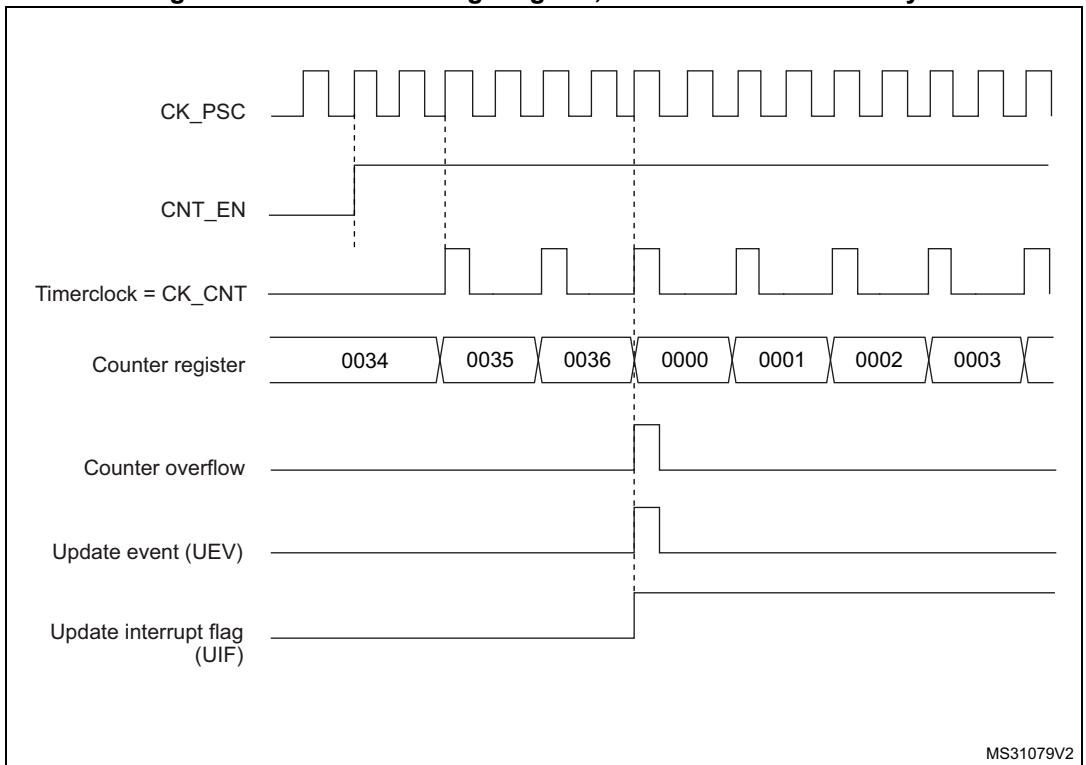
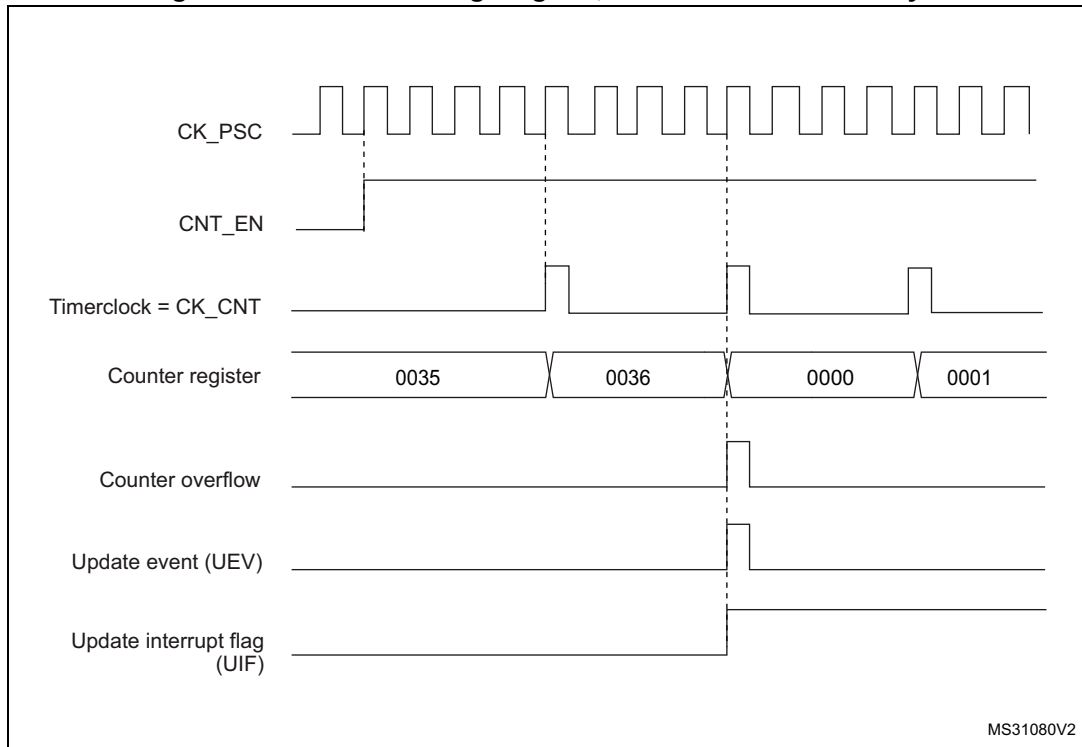
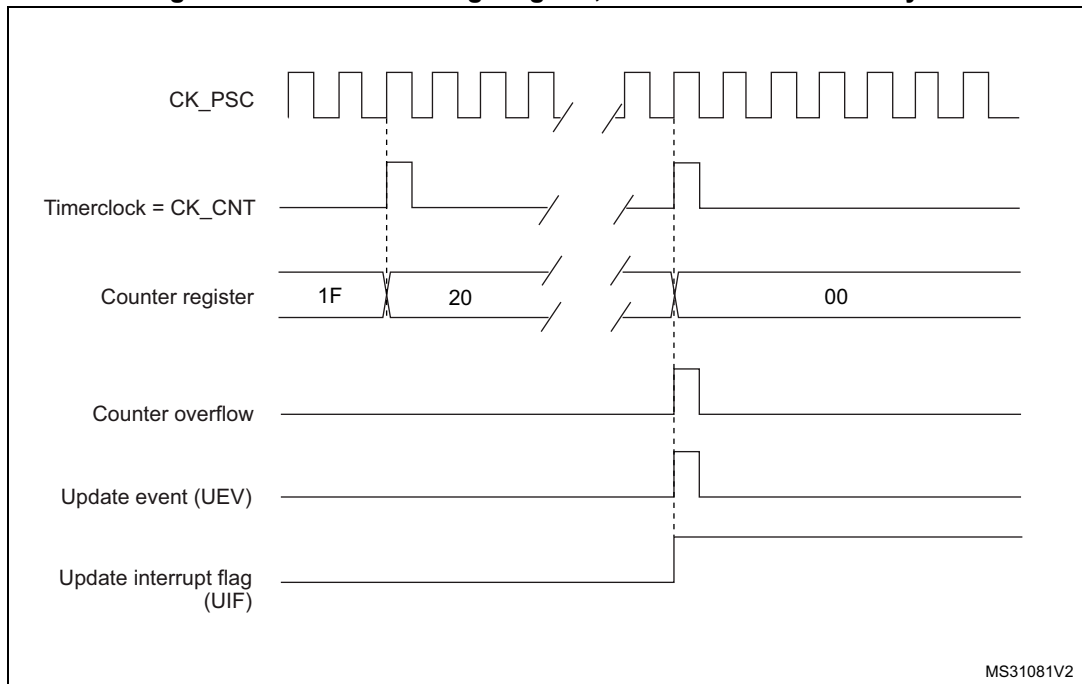


Figure 260. Counter timing diagram, internal clock divided by 4



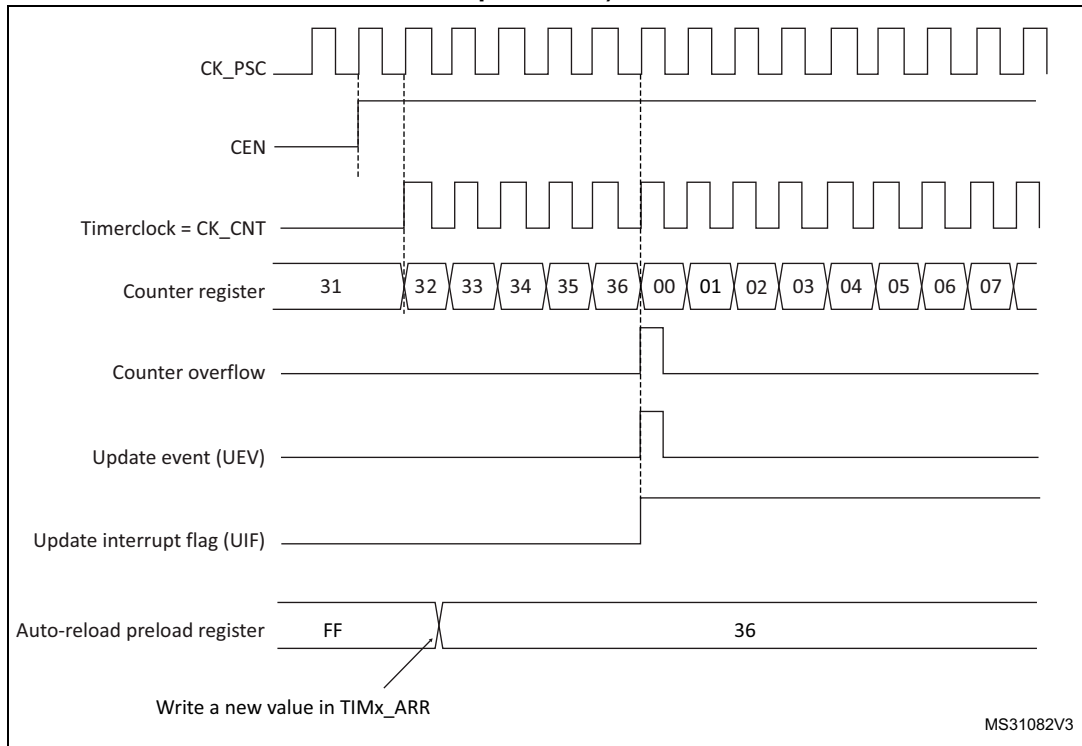
MS31080V2

Figure 261. Counter timing diagram, internal clock divided by N

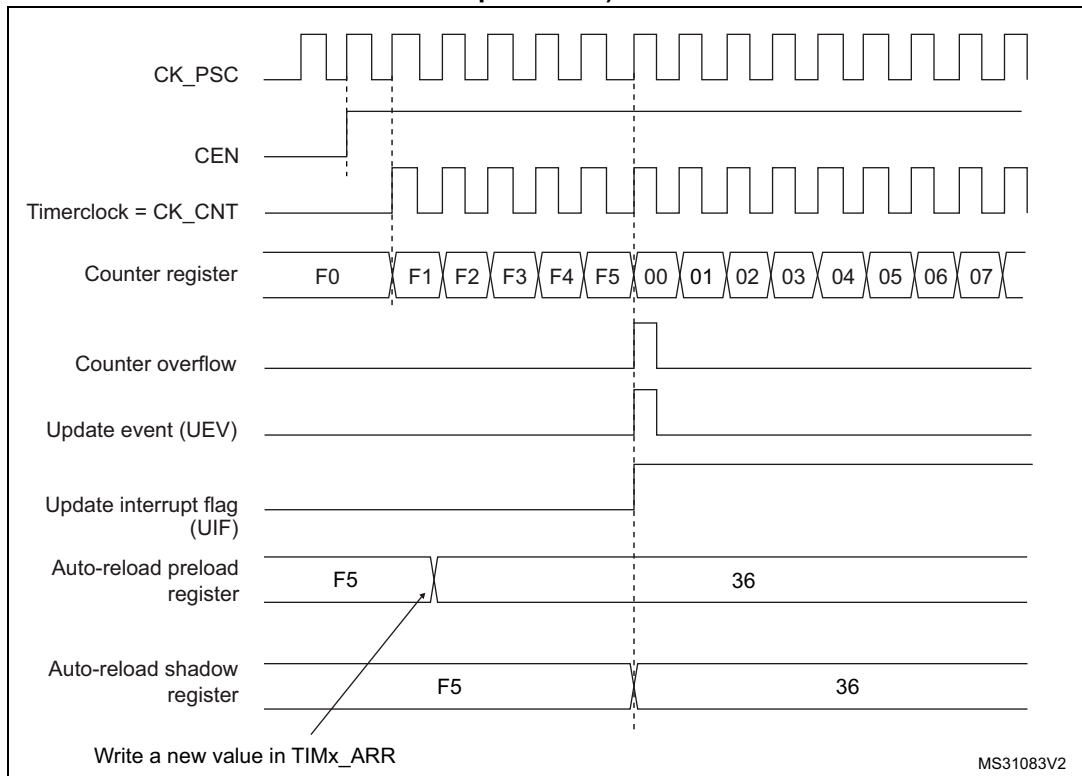


MS31081V2

**Figure 262. Counter timing diagram, update event when ARPE=0 (TIMx\_ARR not preloaded)**



**Figure 263. 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).

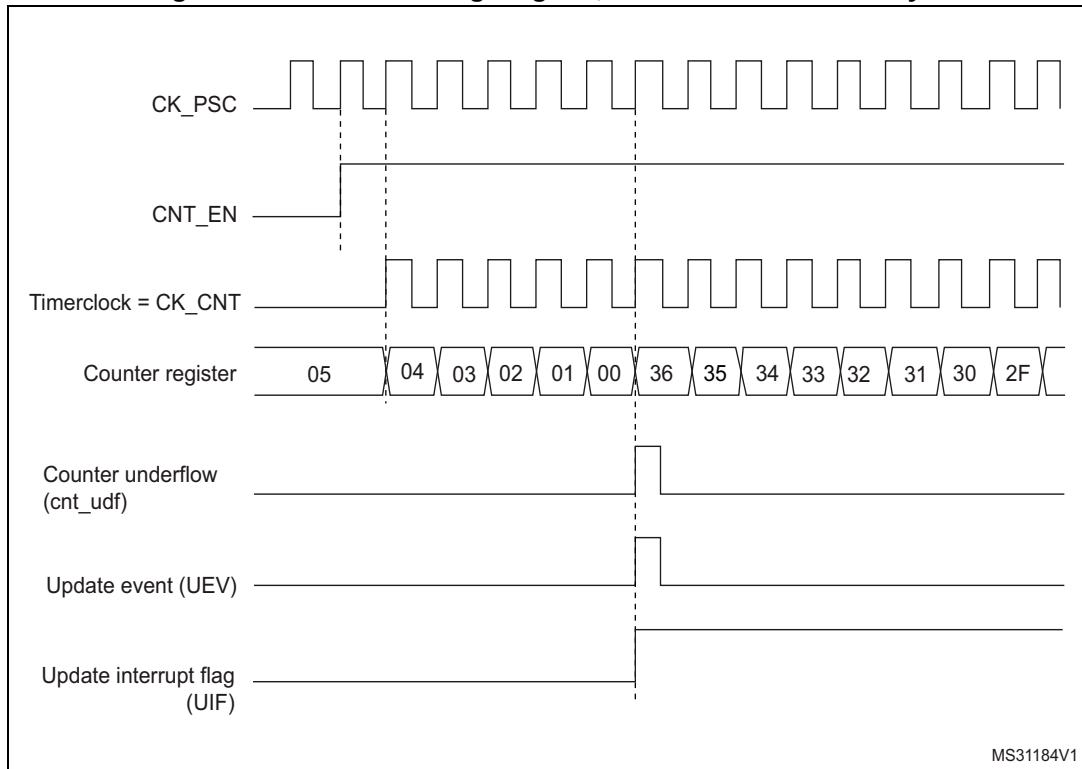
In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on 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.

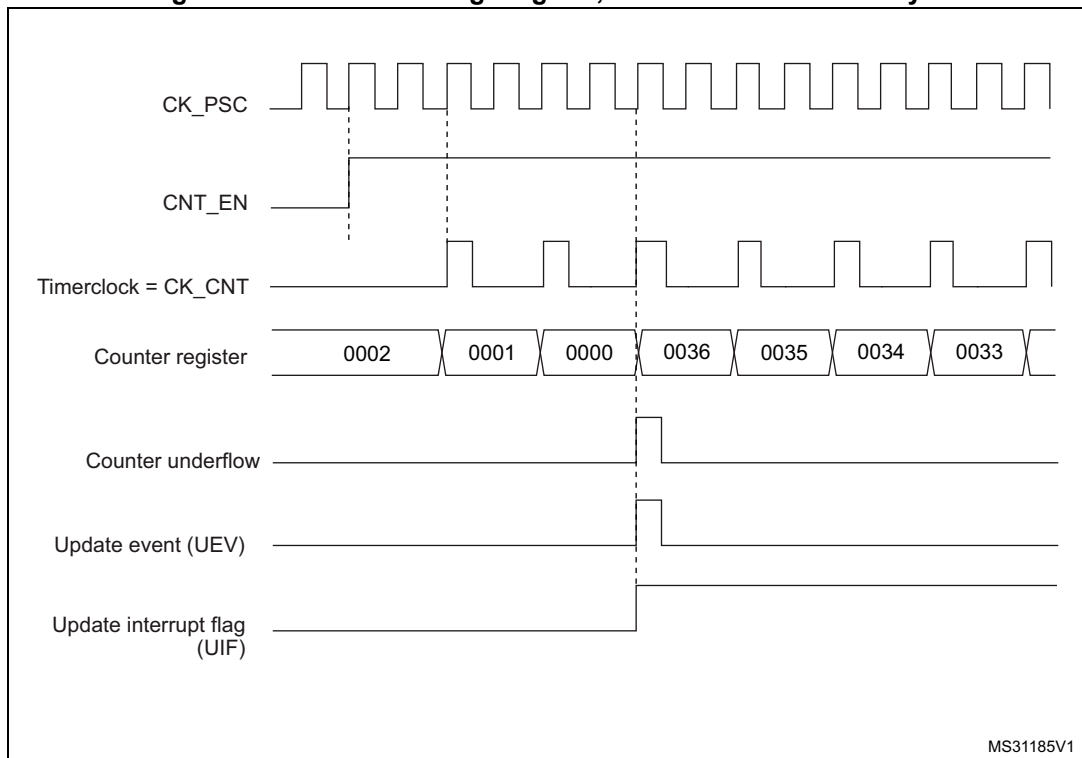
The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR=0x36.

**Figure 264. Counter timing diagram, internal clock divided by 1**



MS31184V1

**Figure 265. Counter timing diagram, internal clock divided by 2**



MS31185V1

Figure 266. Counter timing diagram, internal clock divided by 4

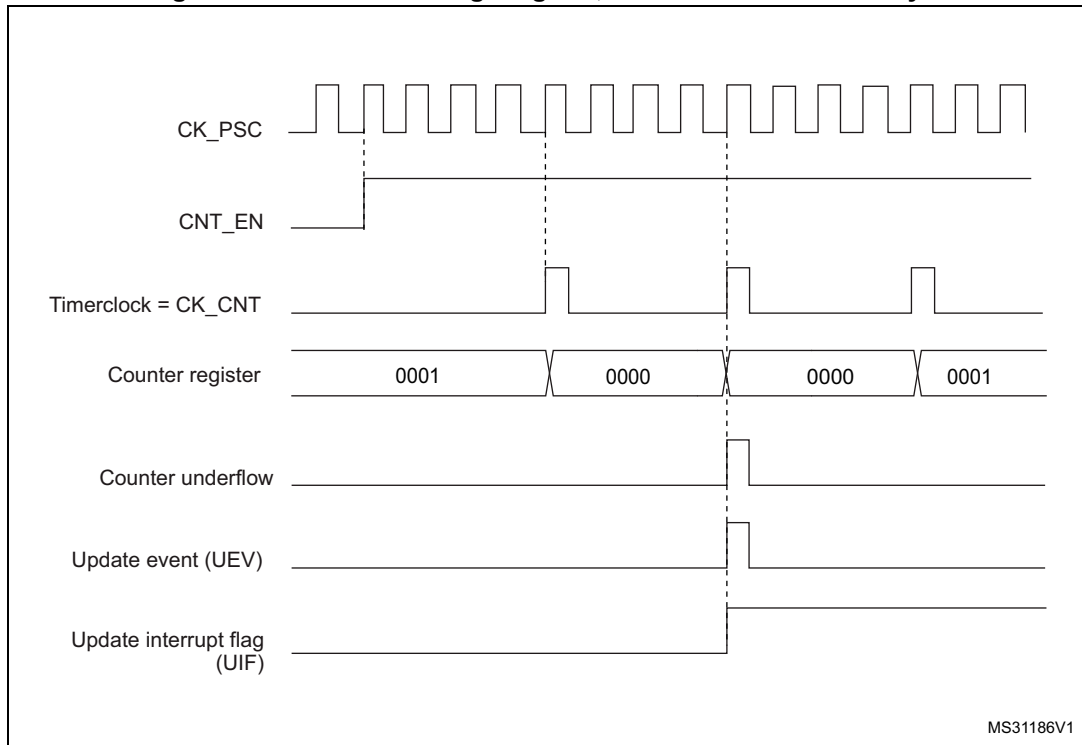
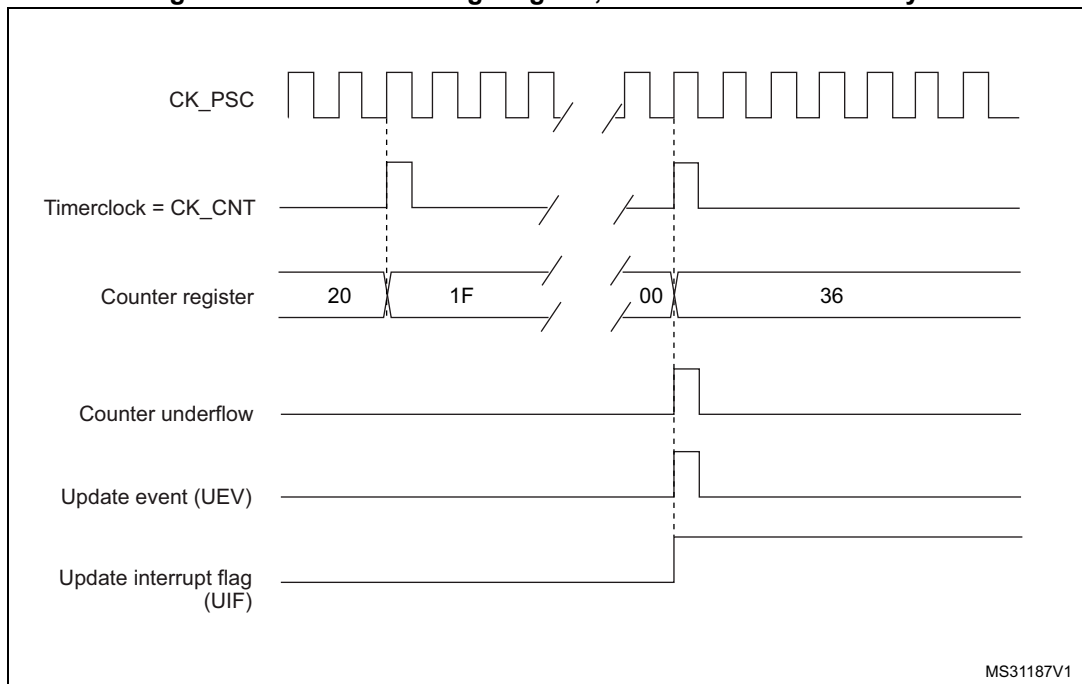
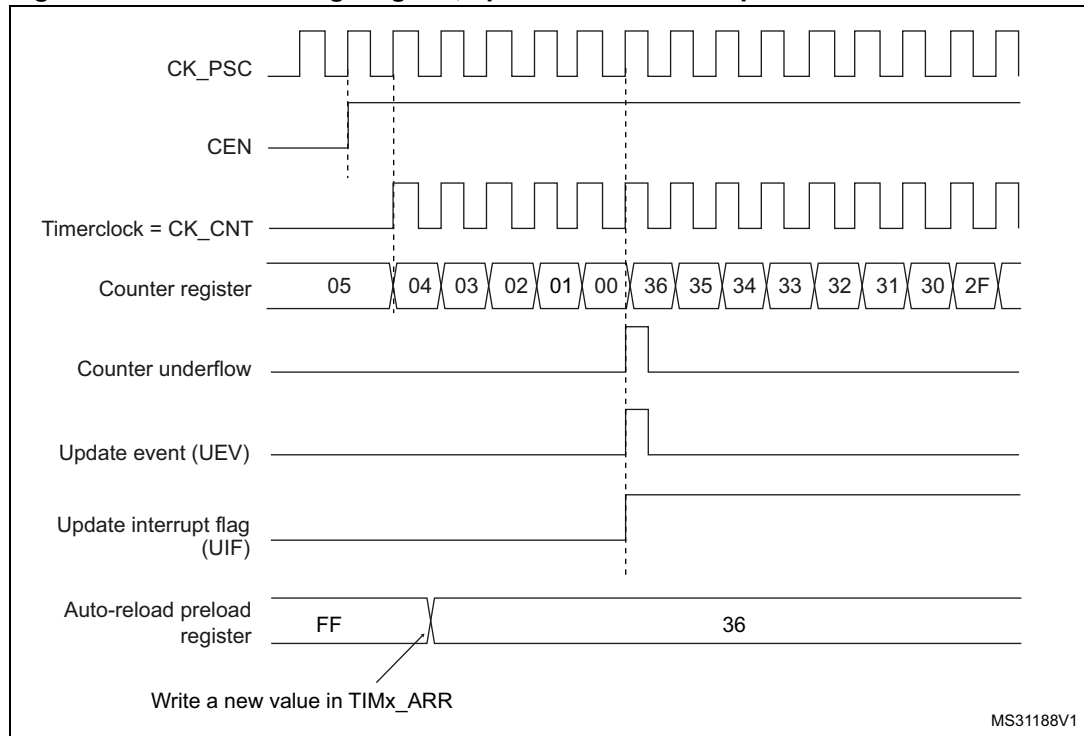


Figure 267. Counter timing diagram, internal clock divided by N



**Figure 268. 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

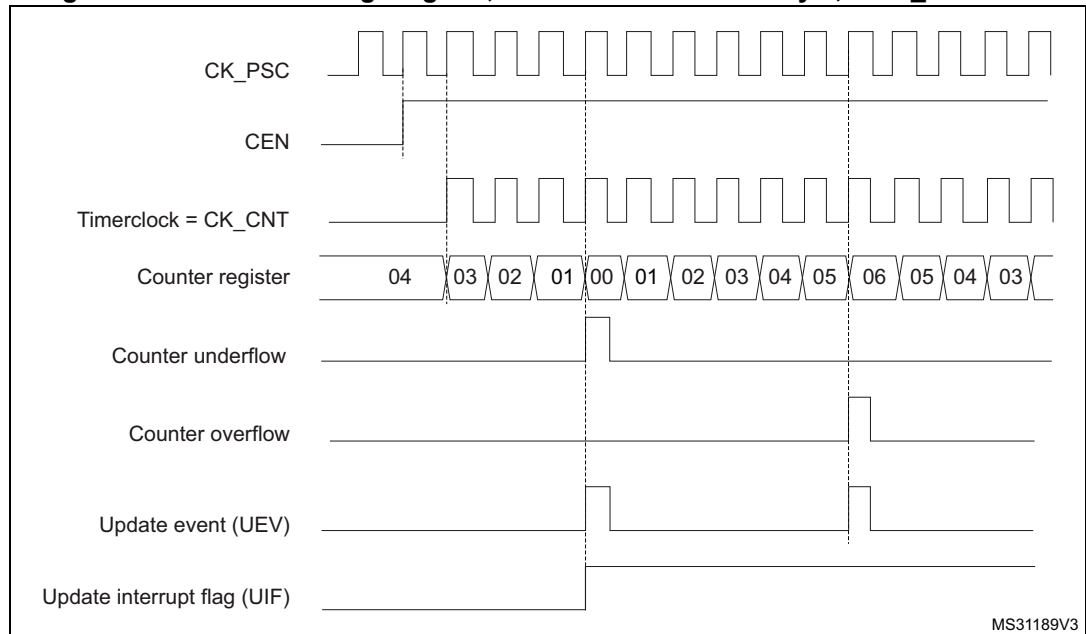
DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

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).

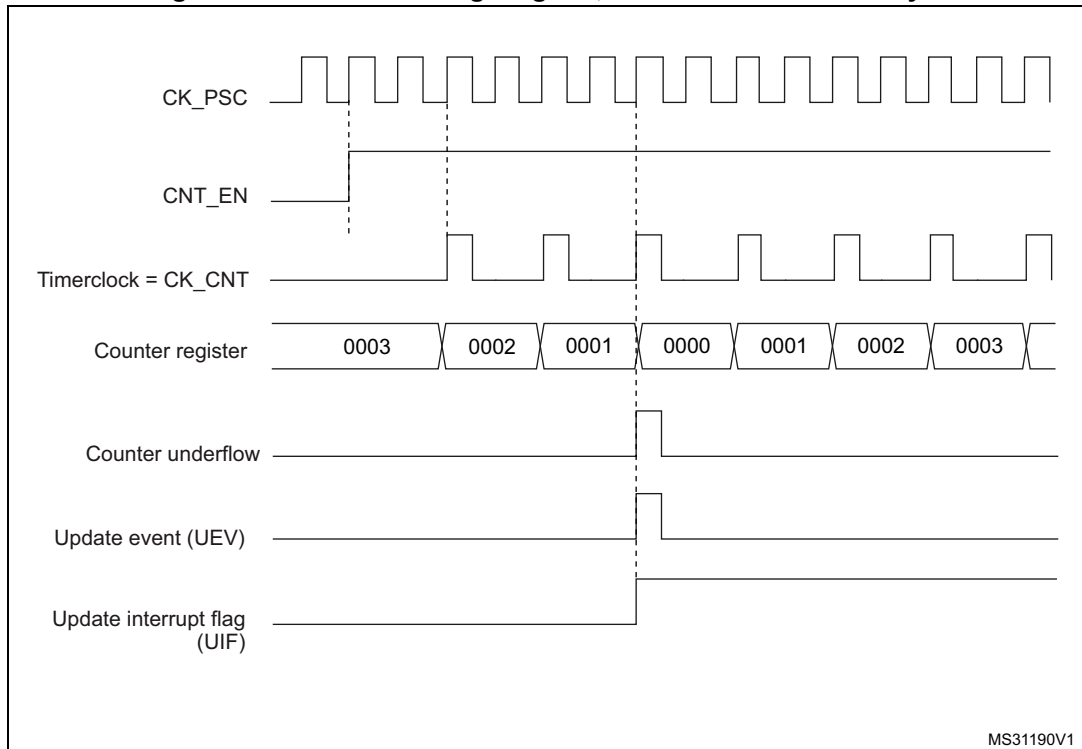
The following figures show some examples of the counter behavior for different clock frequencies.

**Figure 269. Counter timing diagram, internal clock divided by 1, TIMx\_ARR = 0x6**



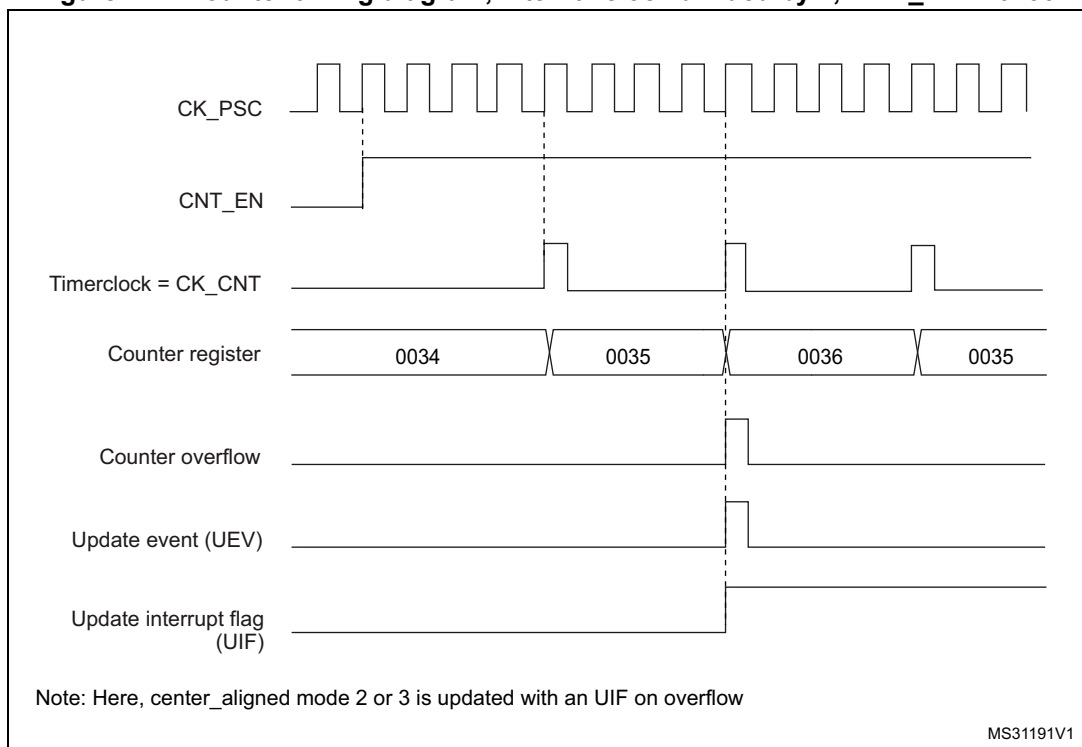
1. Here, center-aligned mode 1 is used (for more details refer to [Section 34.4: TIM1/TIM8 registers](#)).

Figure 270. Counter timing diagram, internal clock divided by 2



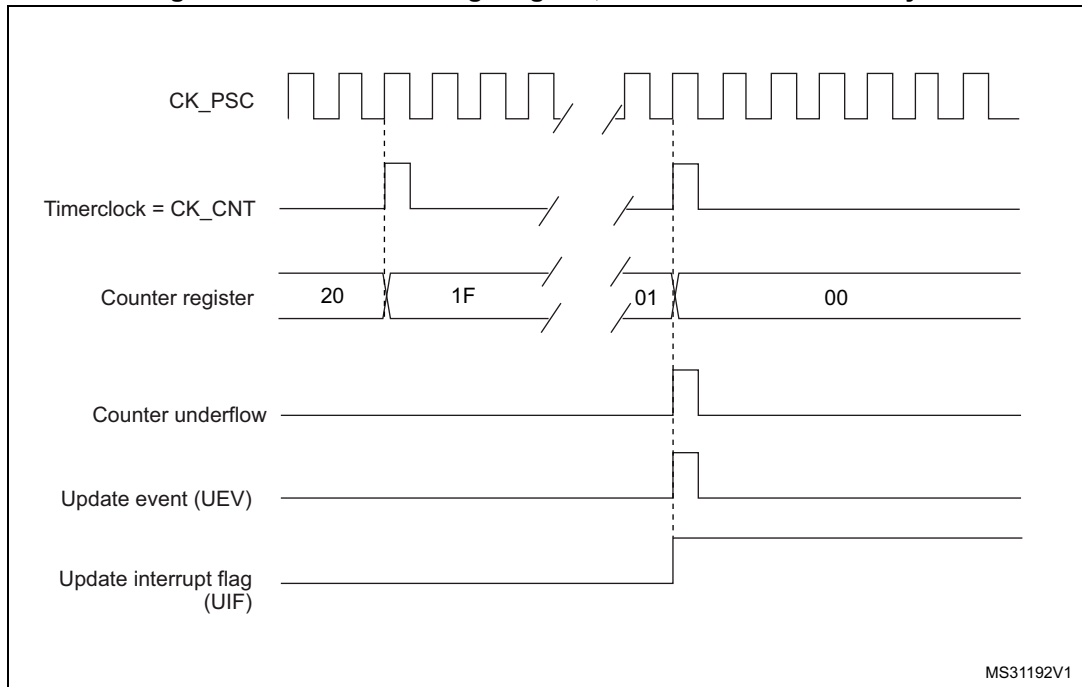
MS31190V1

Figure 271. Counter timing diagram, internal clock divided by 4, TIMx\_ARR=0x36



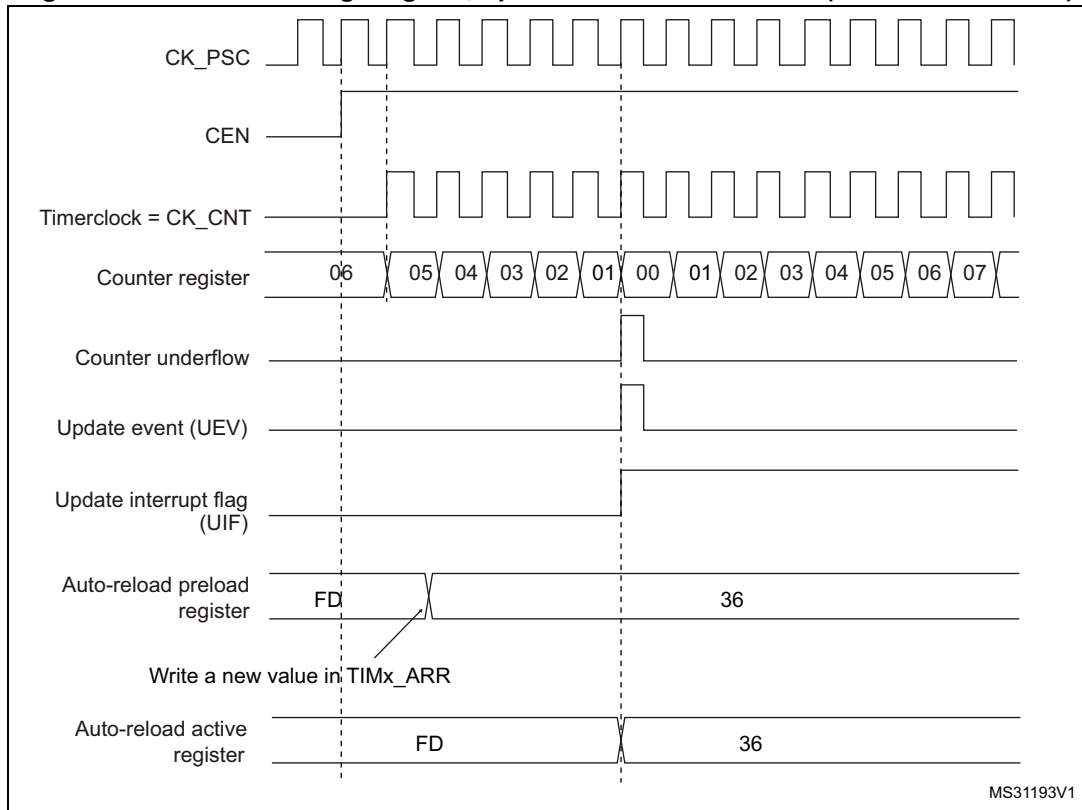
MS31191V1

Figure 272. Counter timing diagram, internal clock divided by N



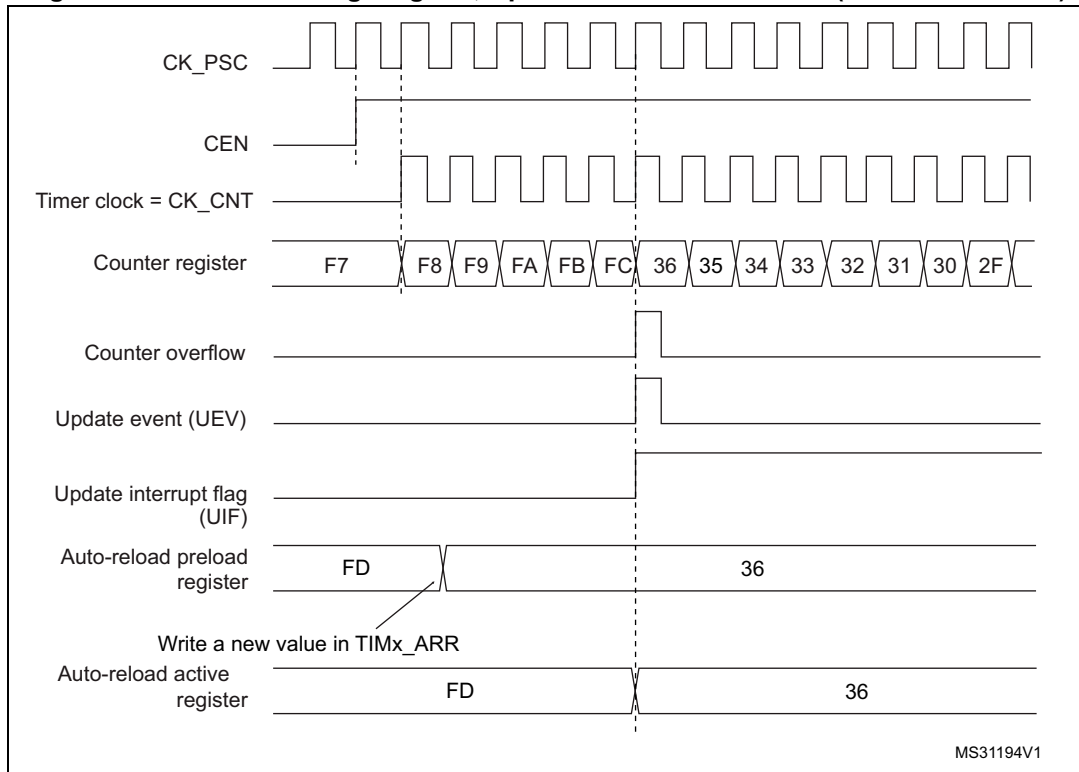
MS31192V1

Figure 273. Counter timing diagram, update event with ARPE=1 (counter underflow)



MS31193V1

Figure 274. Counter timing diagram, Update event with ARPE=1 (counter overflow)



### 34.3.3 Repetition counter

[Section 34.3.1: Time-base unit](#) describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx\_ARR auto-reload register, TIMx\_PSC prescaler register, but also TIMx\_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx\_RCR repetition counter register.

The repetition counter is decremented:

- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is  $2xT_{ck}$ , due to the symmetry of the pattern.

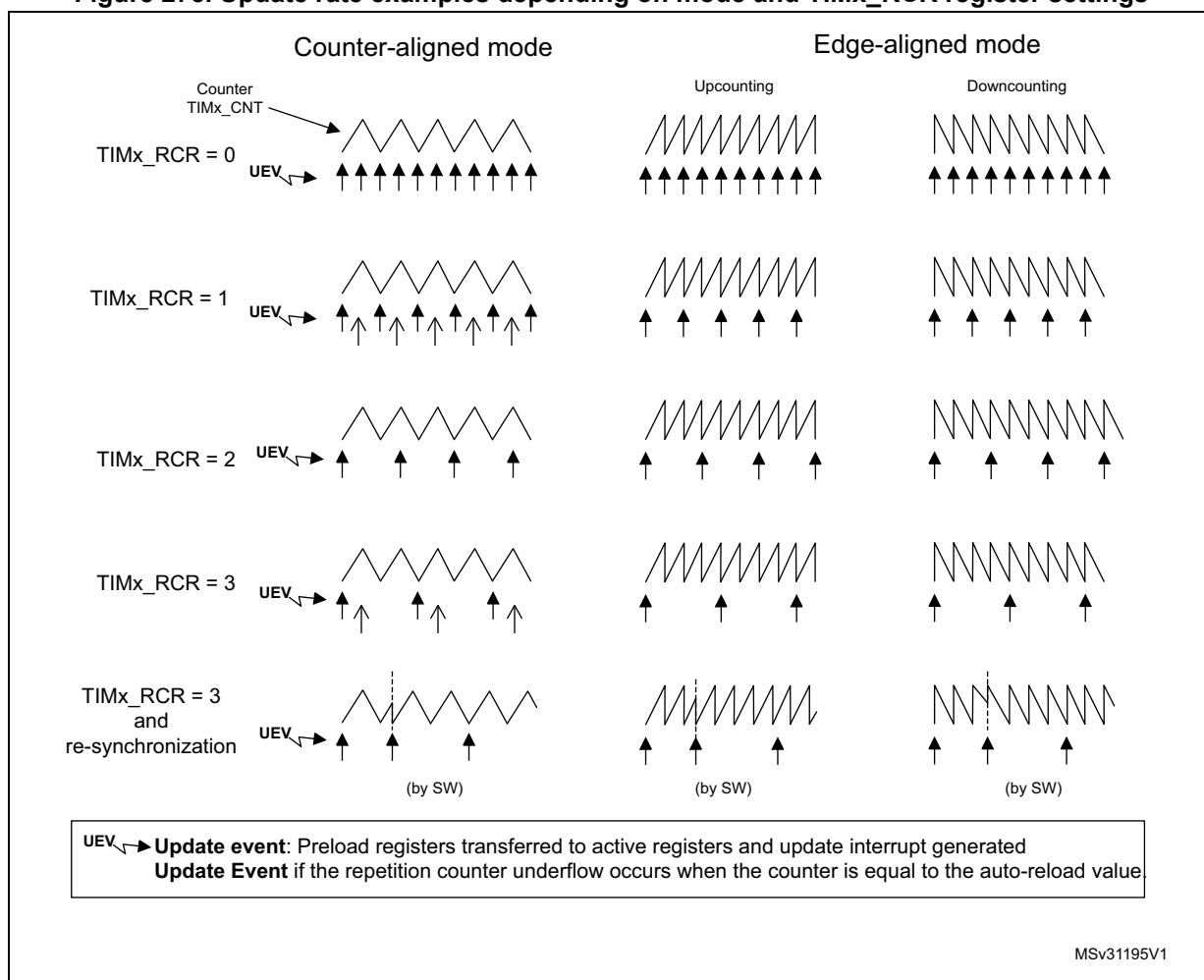
The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx\_RCR register value (refer to [Figure 275](#)). When the update event is generated by software (by setting the UG bit in TIMx\_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx\_RCR register.



In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the overflow. If the RCR was written after launching the counter, the UEV occurs on the underflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

**Figure 275. Update rate examples depending on mode and TIMx\_RCR register settings**



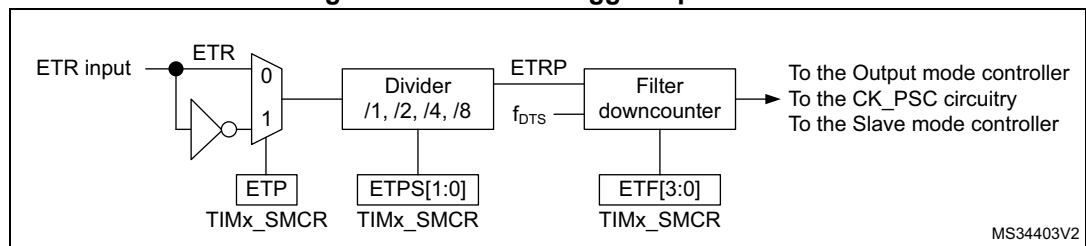
### 34.3.4 External trigger input

The timer features an external trigger input ETR. It can be used as:

- external clock (external clock mode 2, see [Section 34.3.5](#))
- trigger for the slave mode (see [Section 34.3.26](#))
- PWM reset input for cycle-by-cycle current regulation (see [Section 34.3.7](#))

[Figure 276](#) below describes the ETR input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled by the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield.

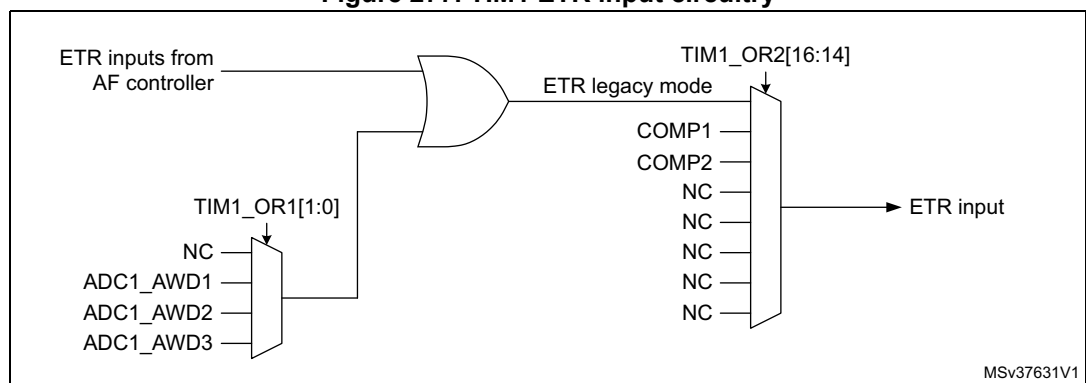
**Figure 276. External trigger input block**



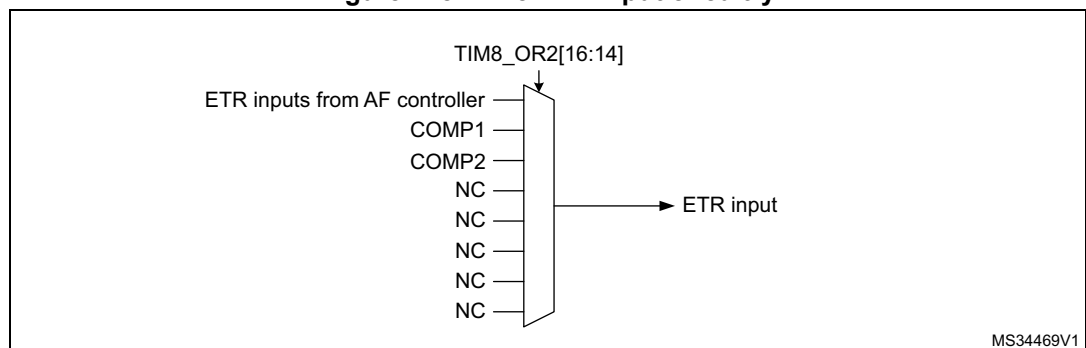
The ETR input comes from multiple sources: input pins (default configuration), comparator outputs and analog watchdogs. The selection is done with:

- the ETRSEL[2:0] bitfield in the TIMx\_OR2 register
- the ETR\_ADC1\_RMP bitfield in the TIMxOR1[1:0] register

**Figure 277. TIM1 ETR input circuitry**



**Figure 278. TIM8 ETR input circuitry**



### 34.3.5 Clock selection

The counter clock can be provided by the following clock sources:

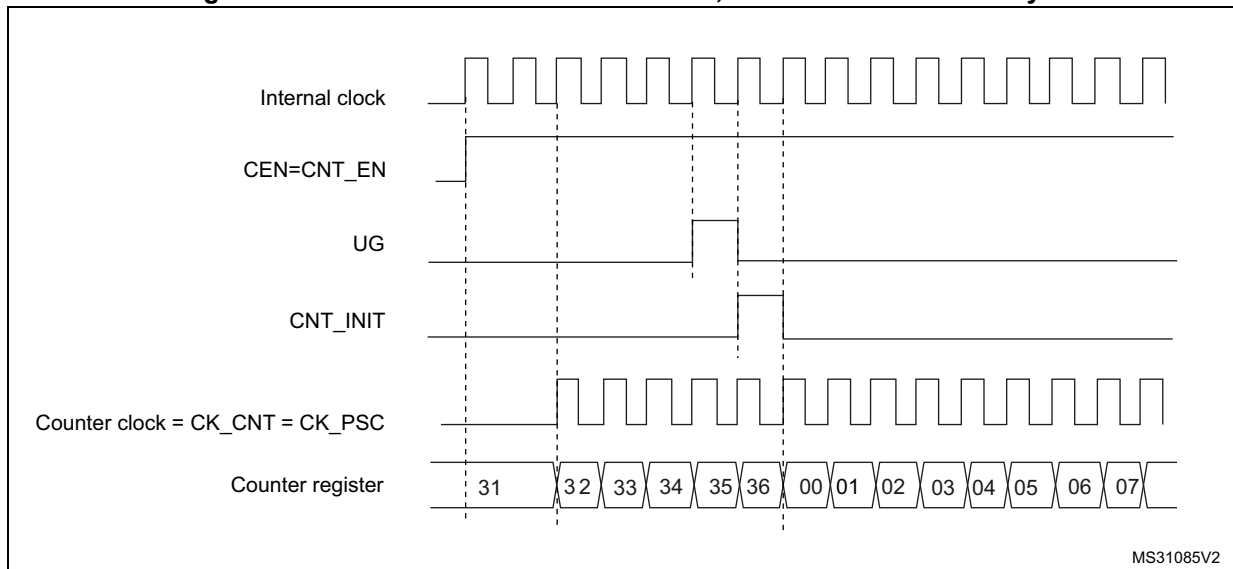
- Internal clock (CK\_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Encoder mode

#### 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 279 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

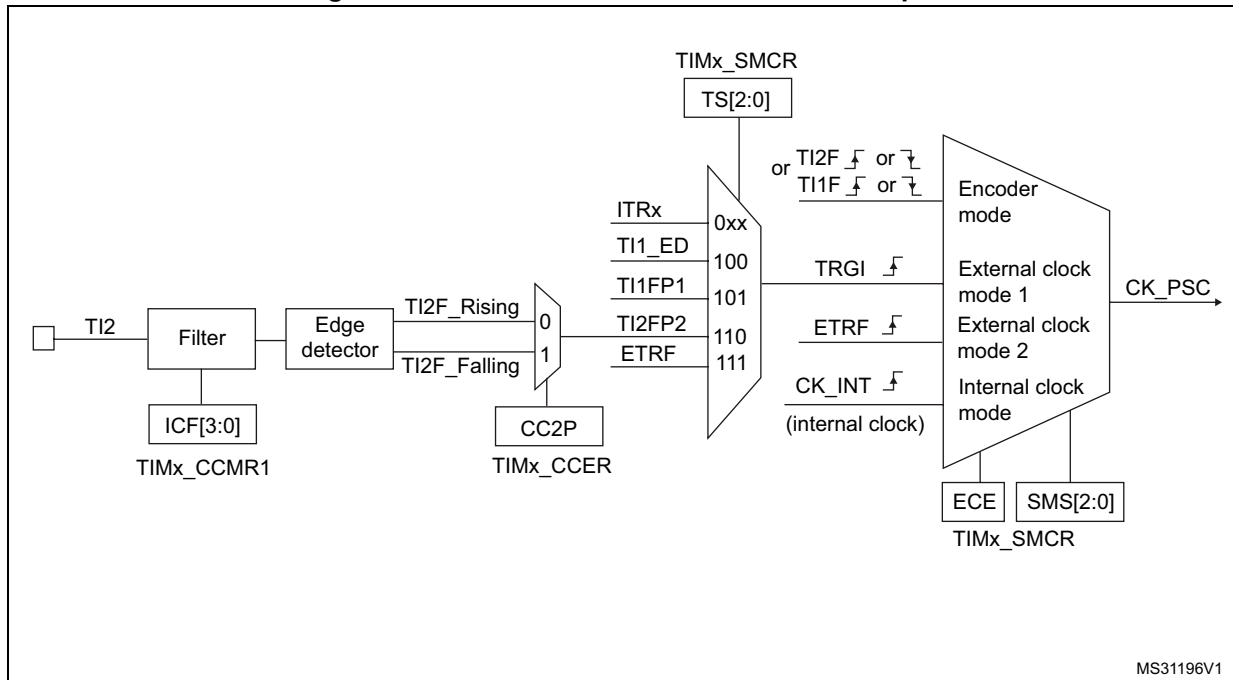
**Figure 279. Control circuit in normal mode, internal clock divided by 1**



#### 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 280. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

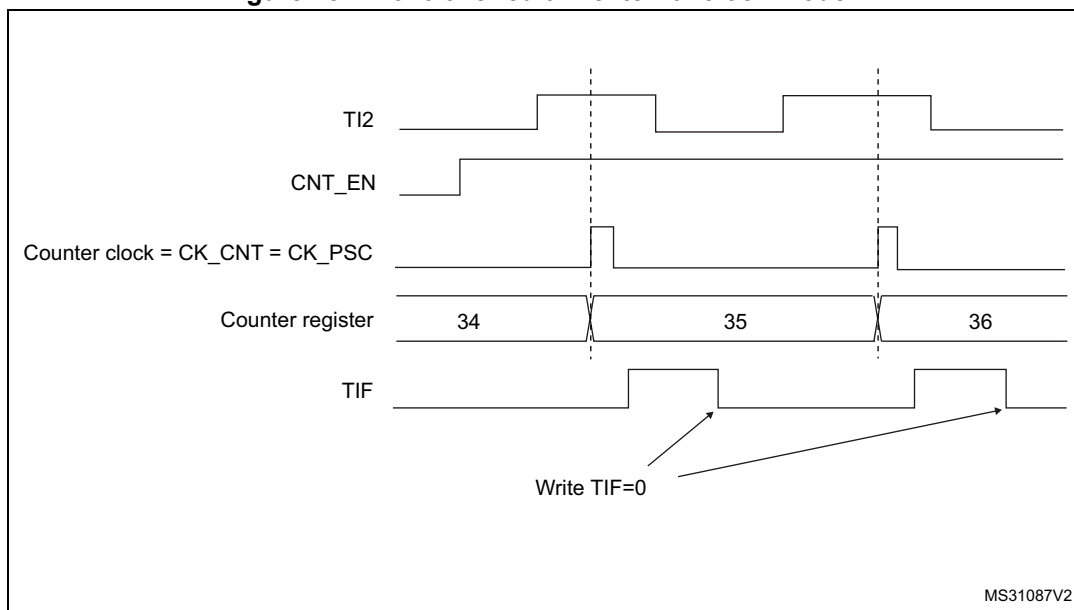
1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx\_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx\_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx\_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx\_SMCR register.
5. Select TI2 as the trigger input source by writing TS=110 in the TIMx\_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx\_CR1 register.

**Note:** *The capture prescaler is not used for triggering, so 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 281. Control circuit in external clock mode 1



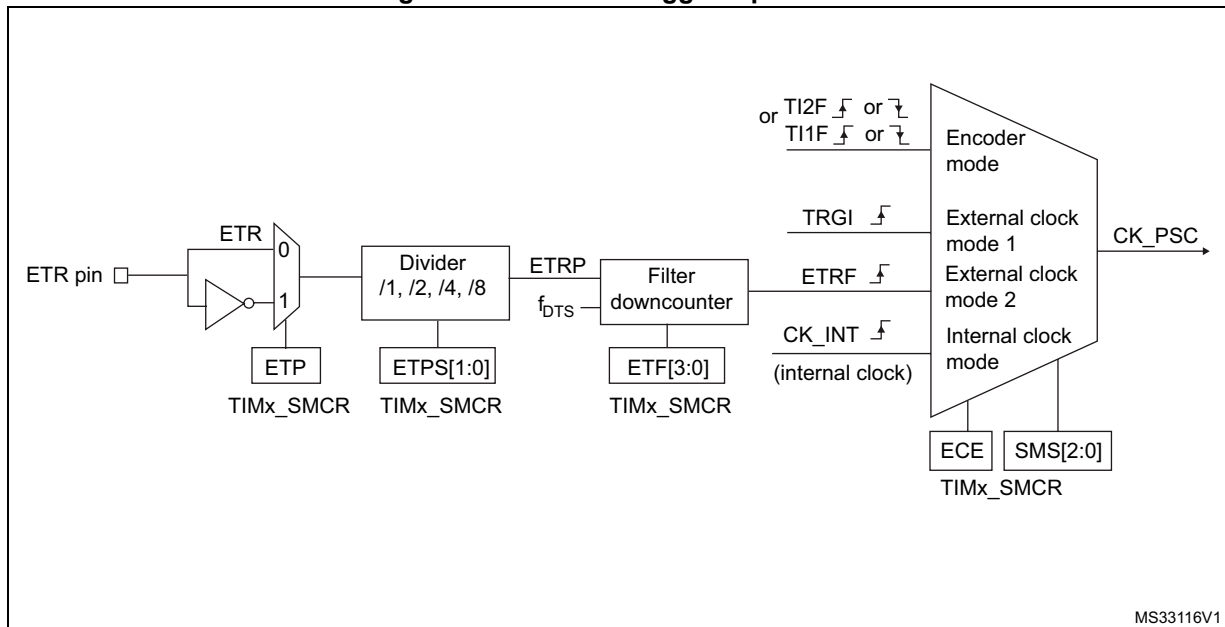
**External clock source mode 2**

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 282](#) gives an overview of the external trigger input block.

Figure 282. External trigger input block



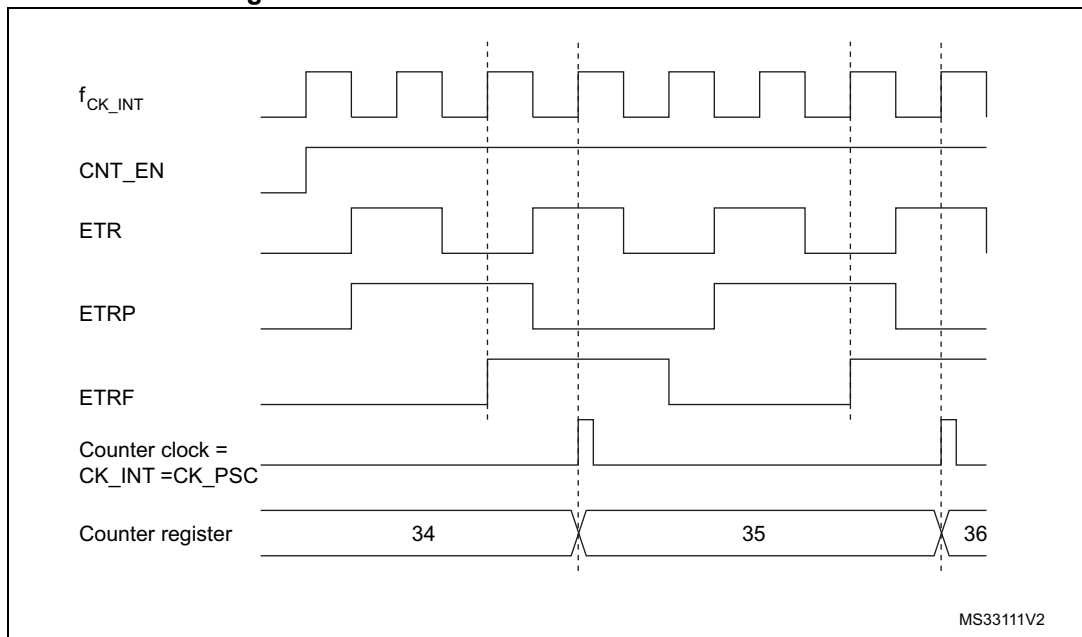
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETRF[3:0]=0000 in the TIMx\_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx\_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx\_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx\_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx\_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

**Figure 283. Control circuit in external clock mode 2**



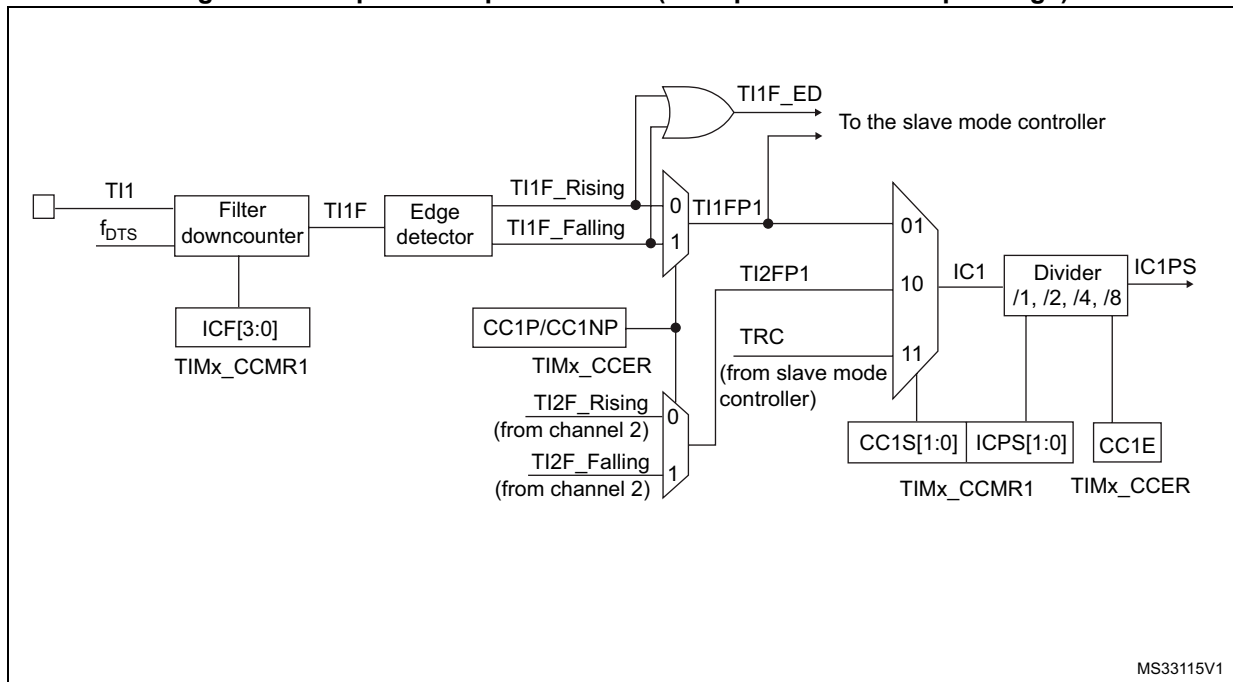
### 34.3.6 Capture/compare channels

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 284 to Figure 287 give an overview of one Capture/Compare channel.

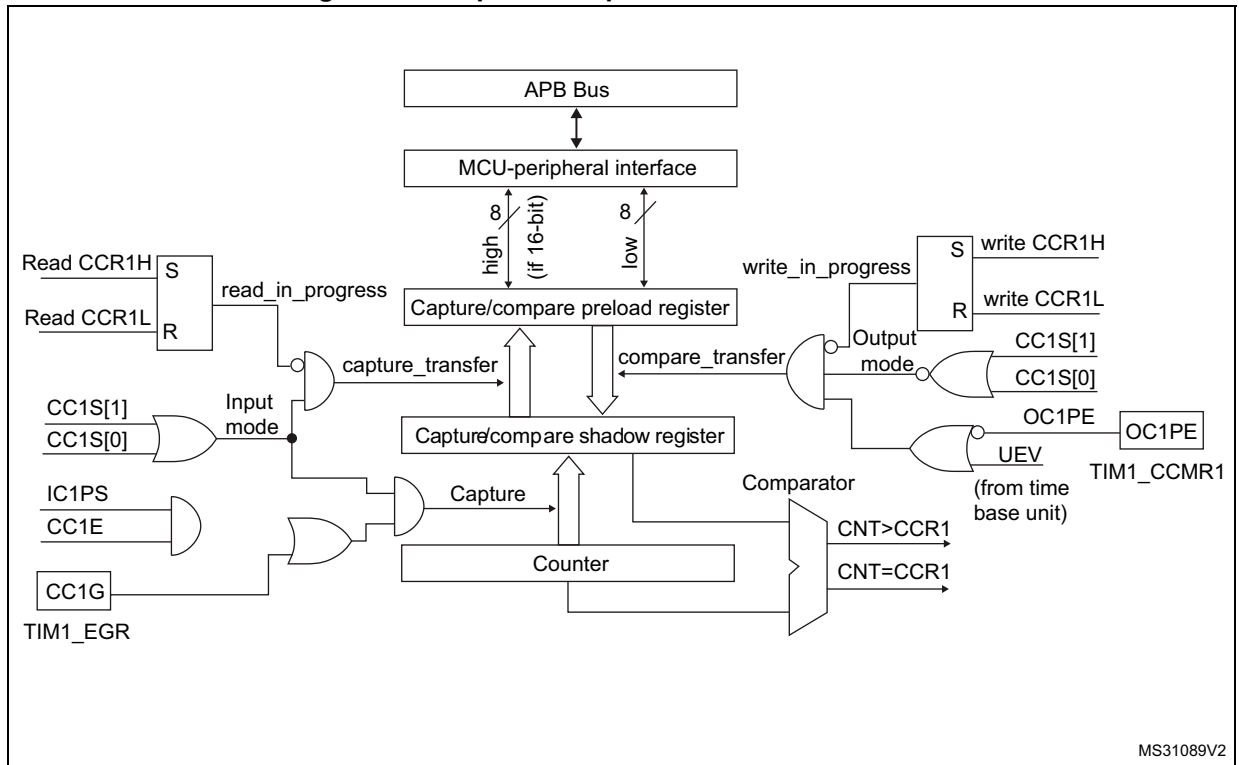
The input stage samples the corresponding Tix input to generate a filtered signal TixF. Then, an edge detector with polarity selection generates a signal (TixFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 284. Capture/compare channel (example: channel 1 input stage)



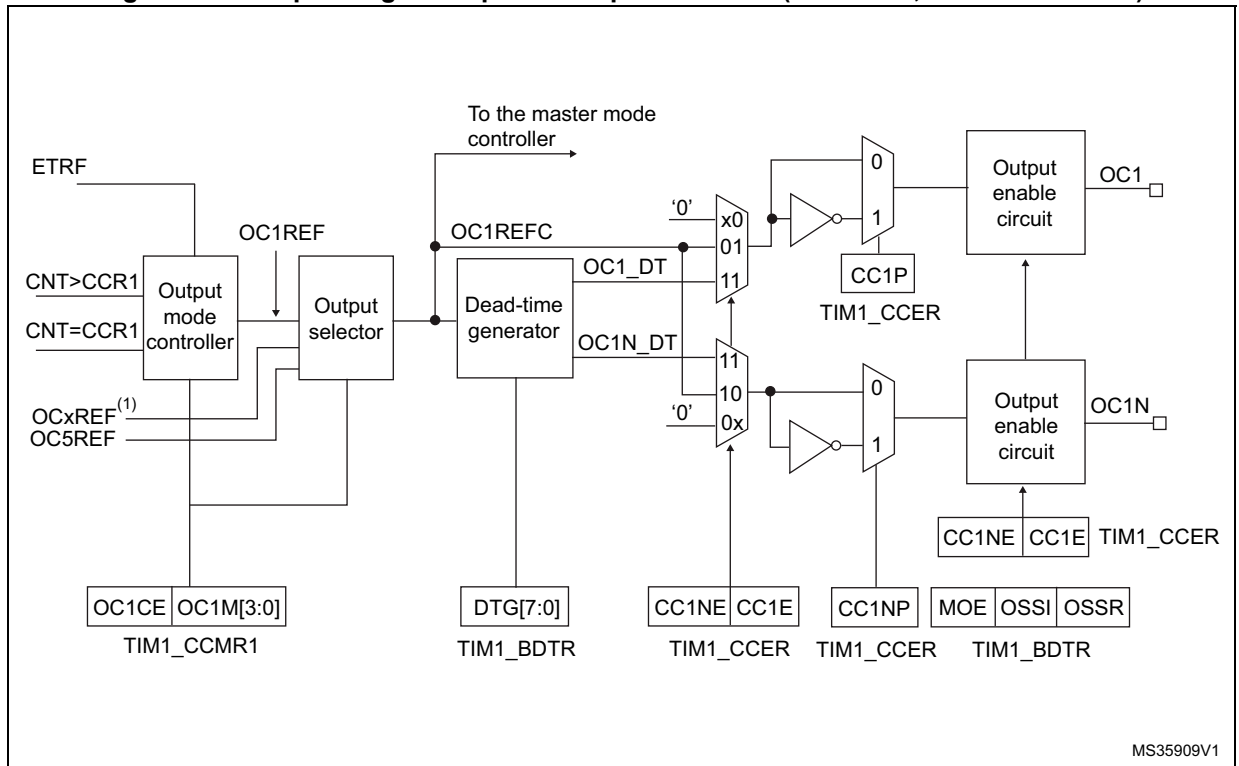
The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 285. Capture/compare channel 1 main circuit



MS31089V2

Figure 286. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3)



MS35909V1

1. OCxREF, where x is the rank of the complementary channel



Figure 287. Output stage of capture/compare channel (channel 4)

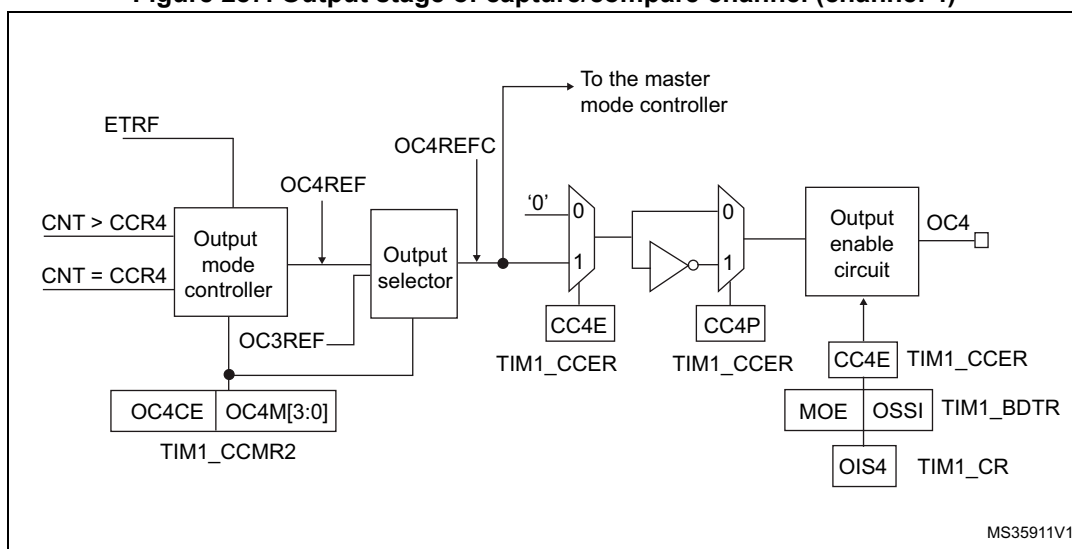
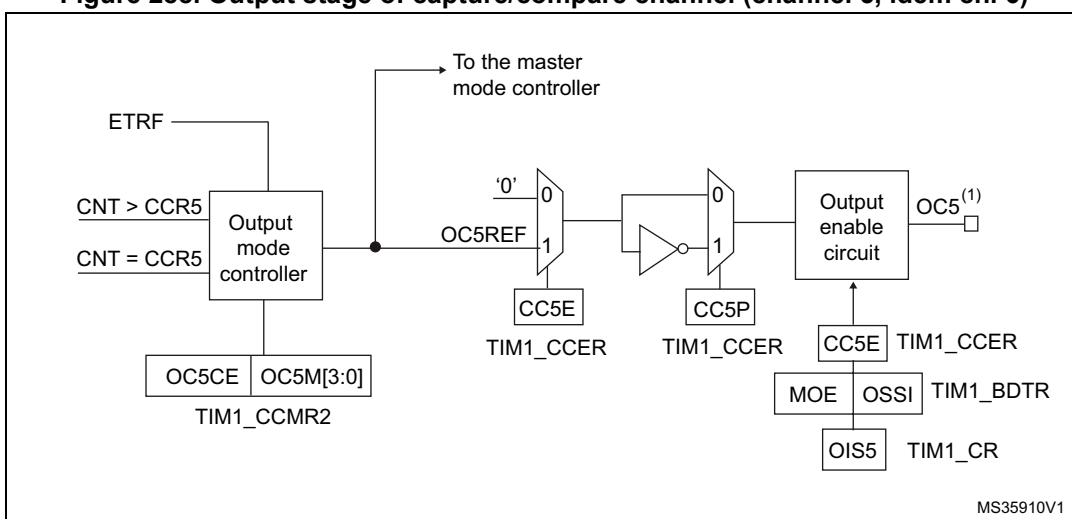


Figure 288. Output stage of capture/compare channel (channel 5, idem ch. 6)



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.

### 34.3.7 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 you write it to '0'.

The following example shows how to capture the counter value in TIMx\_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the active input: TIMx\_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx\_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx\_CCR1 register becomes read-only.
2. Program the input filter duration you need with respect to the signal you connect to the timer (when the input is one of the TIx (ICxF bits in the TIMx\_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at  $f_{DTS}$  frequency). Then write IC1F bits to 0011 in the TIMx\_CCMR1 register.
3. Select the edge of the active transition on the TI1 channel by writing CC1P and CC1NP bits to 0 in the TIMx\_CCER register (rising edge in this case).
4. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx\_CCMR1 register).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx\_CCER register.
6. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx\_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx\_DIER register.

When an input capture occurs:

- The TIMx\_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

*Note:* IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx\_EGR register.

### 34.3.8 PWM input mode

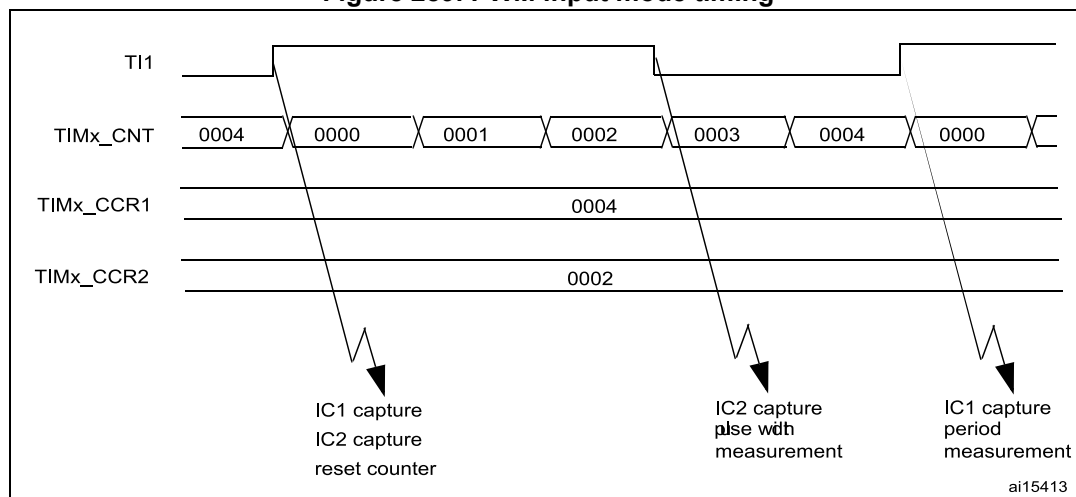
This mode is a particular case of input capture mode. The procedure is the same except:

- 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 active input for TIMx\_CCR1: write the CC1S bits to 01 in the TIMx\_CCMR1 register (TI1 selected).
2. Select the active polarity for TI1FP1 (used both for capture in TIMx\_CCR1 and counter clear): write the CC1P and CC1NP bits to '0' (active on rising edge).
3. Select the active input for TIMx\_CCR2: write the CC2S bits to 10 in the TIMx\_CCMR1 register (TI1 selected).
4. Select the active polarity for TI1FP2 (used for capture in TIMx\_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP='10' (active on falling edge).
5. Select the valid trigger input: write the TS bits to 101 in the TIMx\_SMCR register (TI1FP1 selected).
6. Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx\_SMCR register.
7. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx\_CCER register.

**Figure 289. PWM input mode timing**



### 34.3.9 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.

### 34.3.10 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 microcontroller (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx\_CCMRx register) and the output polarity (CCxP bit in the TIMx\_CCER register). The output pin can keep its level (OCxM=0000), be set active (OCxM=0001), be set inactive (OCxM=0010) or can toggle (OCxM=0011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx\_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCxIE bit in the TIMx\_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx\_DIER register, CCDS bit in the TIMx\_CR2 register for the DMA request selection).

The TIMx\_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx\_CCMRx register.

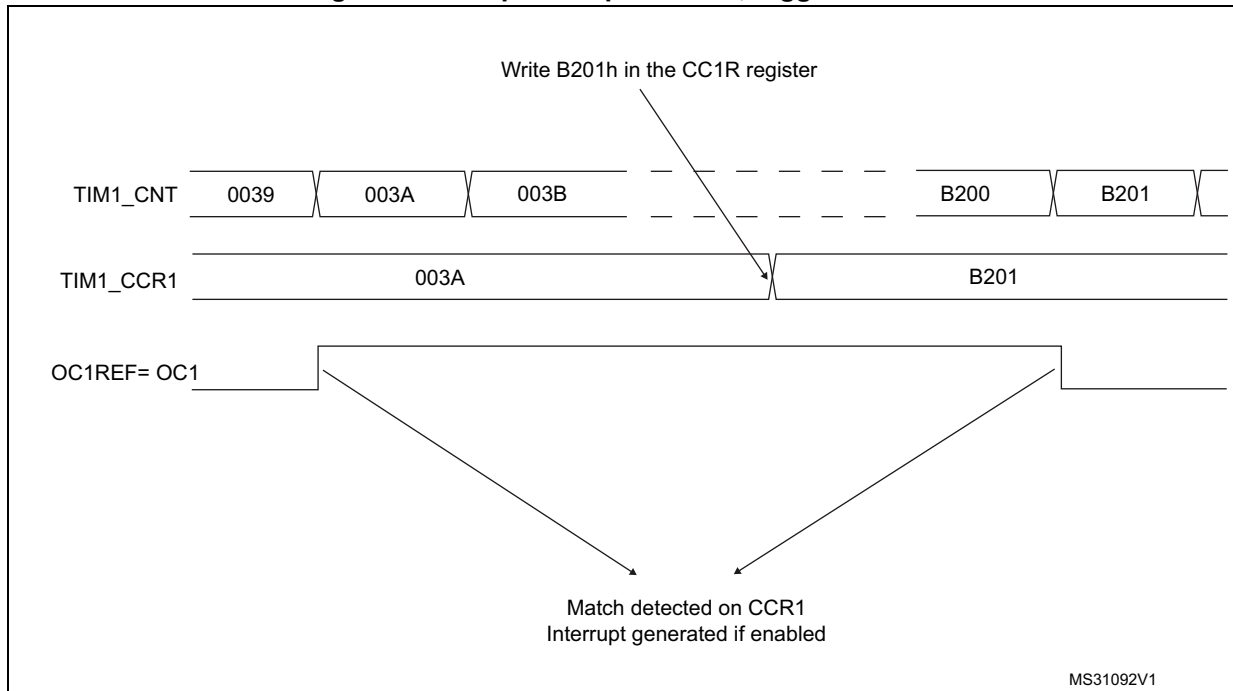
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

#### Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx\_ARR and TIMx\_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
  - Write OCxM = 0011 to toggle OCx output pin when CNT matches CCRx
  - Write OCxPE = 0 to disable preload register
  - Write CCxP = 0 to select active high polarity
  - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx\_CR1 register.

The TIMx\_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx\_CCRx shadow register is updated only at the next update event UEV). An example is given in [Figure 290](#).

Figure 290. Output compare mode, toggle on OC1



### 34.3.11 PWM mode

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx\_ARR register and a duty cycle determined by the value of the TIMx\_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '0110' (PWM mode 1) or '0111' (PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx\_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx\_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx\_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx\_CCER and TIMx\_BDTR registers). Refer to the TIMx\_CCER register description for more details.

In PWM mode (1 or 2), TIMx\_CNT and TIMx\_CCRx are always compared to determine whether  $TIMx\_CCRx \leq TIMx\_CNT$  or  $TIMx\_CNT \leq TIMx\_CCRx$  (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx\_CR1 register.

**PWM edge-aligned mode**

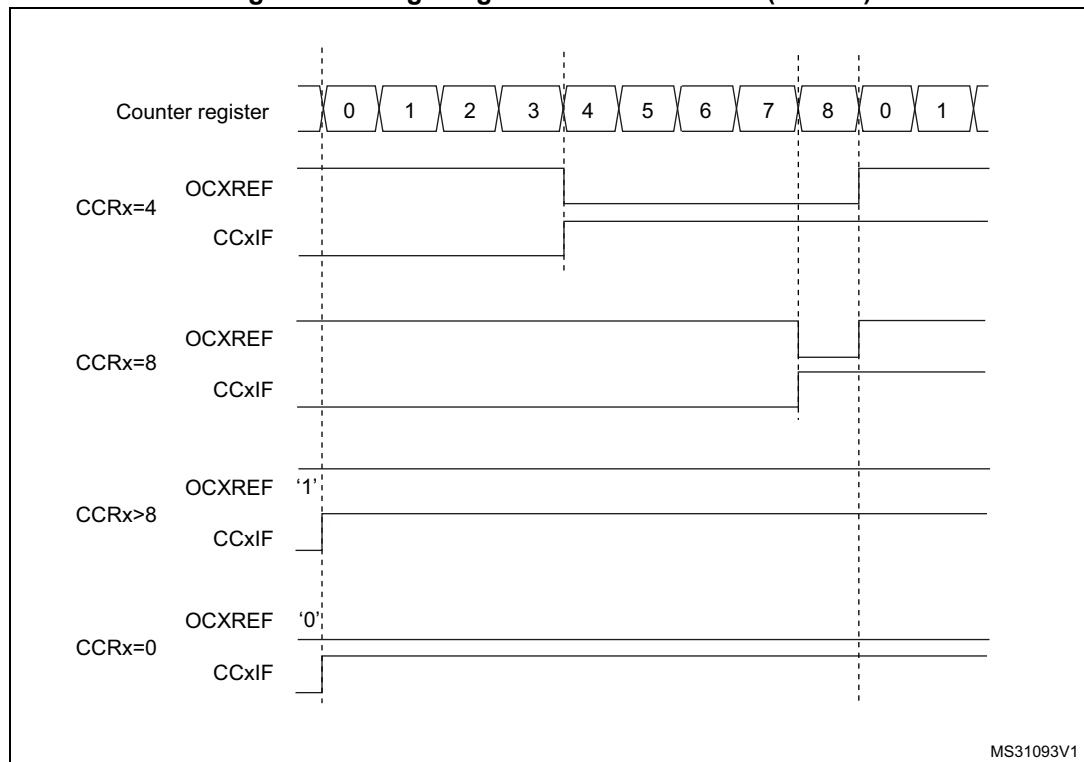
- Upcounting configuration

Upcounting is active when the DIR bit in the TIMx\_CR1 register is low. Refer to the [Upcounting mode on page 1087](#).

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 291](#) shows some edge-aligned PWM waveforms in an example where TIMx\_ARR=8.

**Figure 291. 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 1091](#)

In PWM mode 1, the reference signal OCxRef is low as long as TIMx\_CNT > TIMx\_CCRx else it becomes high. If the compare value in TIMx\_CCRx is greater than the auto-reload value in TIMx\_ARR, then OCxREF is held at '1'. 0% PWM is not possible in this mode.

**PWM center-aligned mode**

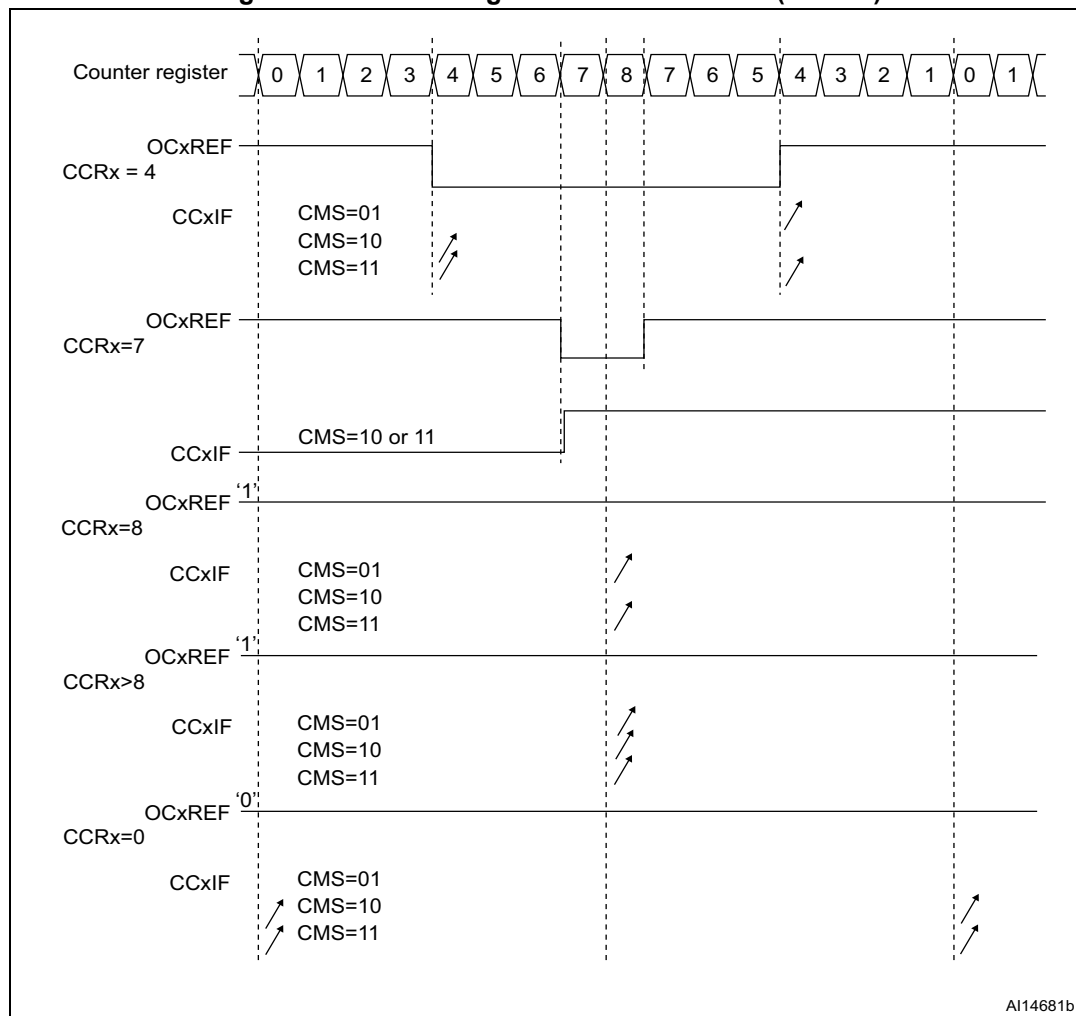
Center-aligned mode is active when the CMS bits in TIMx\_CR1 register are different from '00' (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the

TIMx\_CR1 register is updated by hardware and must not be changed by software. Refer to the [Center-aligned mode \(up/down counting\) on page 1094](#).

Figure 292 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 292. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIMx\_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if you write a value in the counter that is greater than the auto-reload value (TIMx\_CNT > TIMx\_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if you write 0 or write the TIMx\_ARR value in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx\_EGR register) just before starting the counter and not to write the counter while it is running.

### 34.3.12 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.

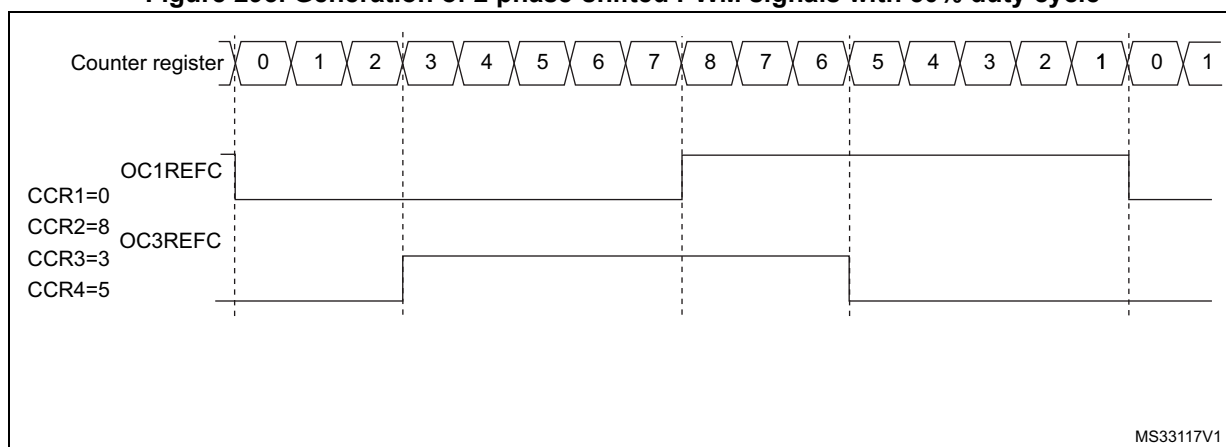
*Note:* The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

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 293](#) 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 293. Generation of 2 phase-shifted PWM signals with 50% duty cycle



MS33117V1

### 34.3.13 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.

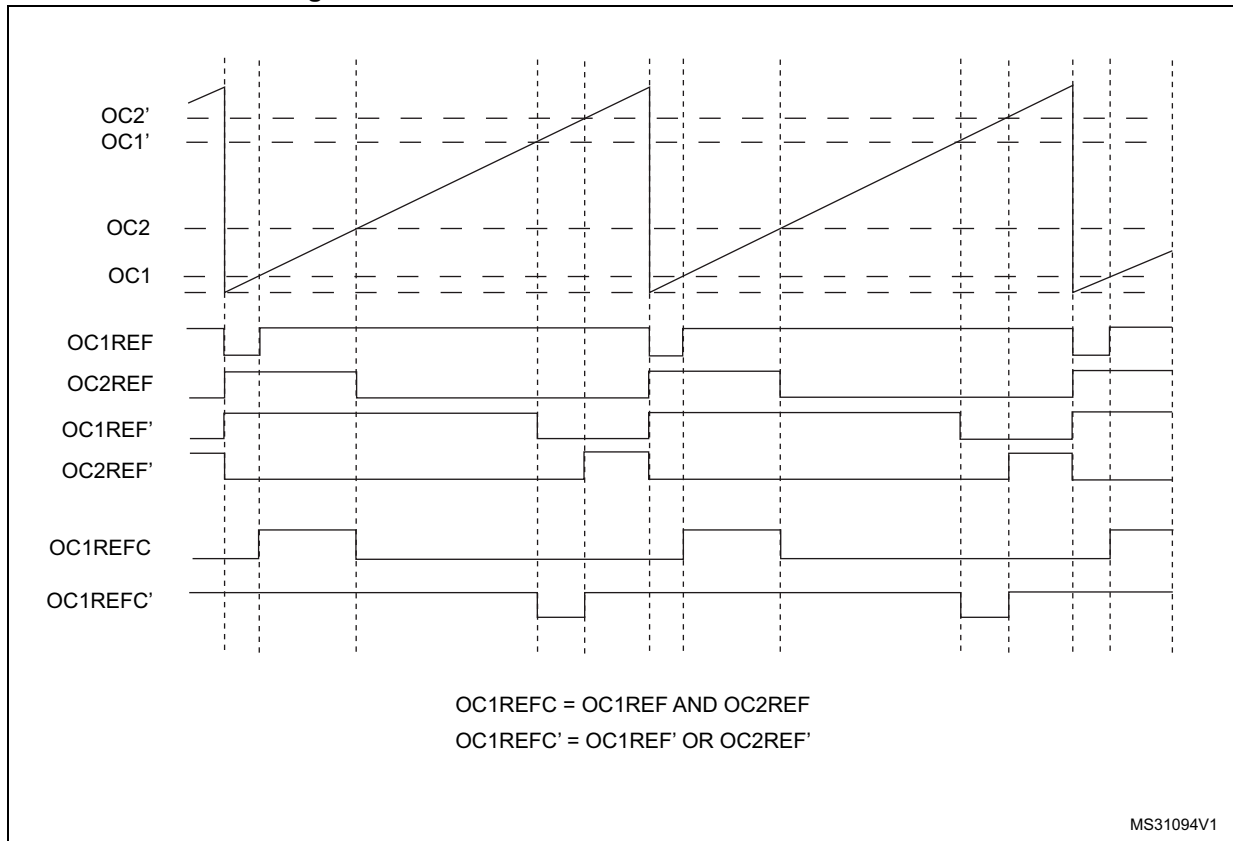
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

*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 294 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 294. Combined PWM mode on channel 1 and 3



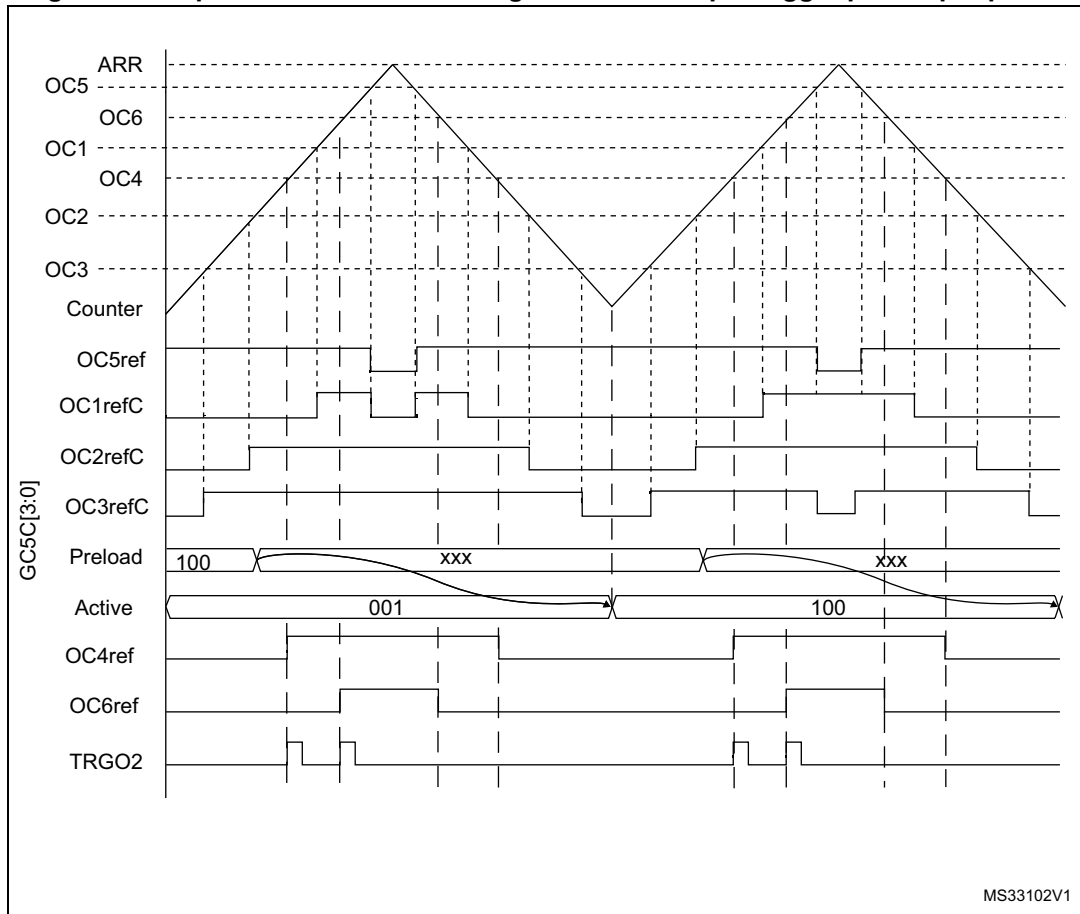
### 34.3.14 Combined 3-phase PWM mode

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 295. 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 34.3.27: ADC synchronization](#) for more details.

### 34.3.15 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1/TIM8) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and you have to adjust it depending on the devices you have connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

You can select the polarity of the outputs (main output OCx or complementary OCxN) independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx\_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx\_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx\_BDTR and TIMx\_CR2 registers. Refer to [Table 218: Output control bits for complementary OCx and OCxN channels with break feature on page 1163](#) for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

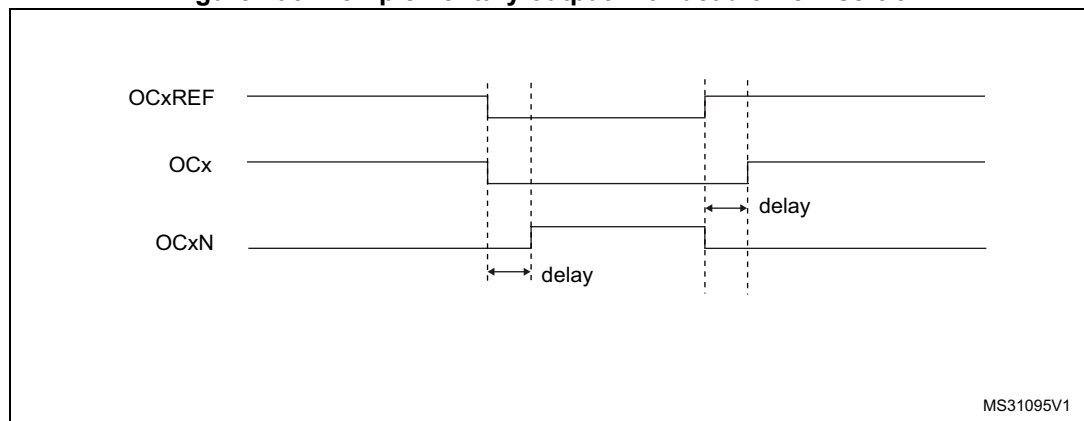
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- 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.

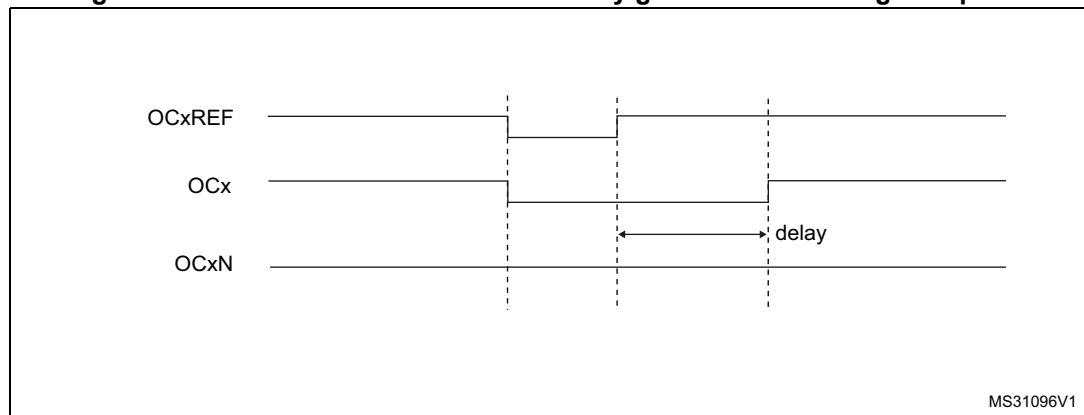
If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

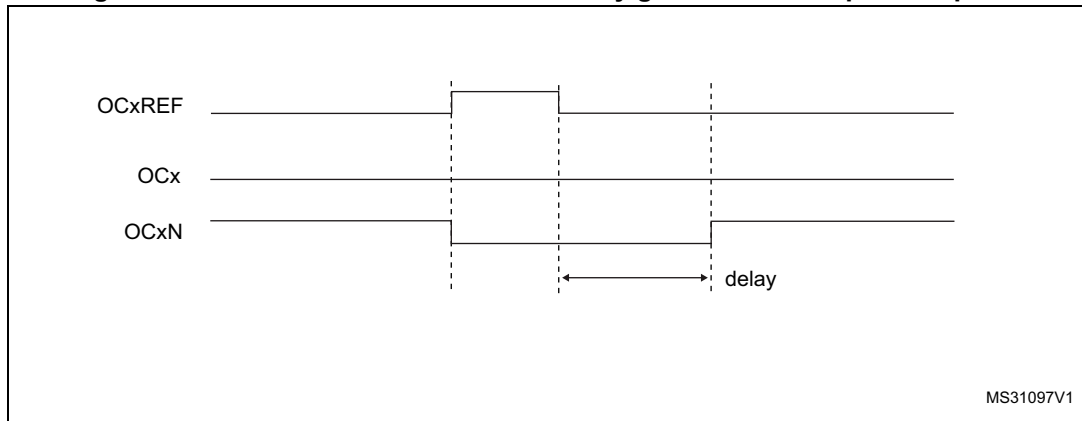
The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

**Figure 296. Complementary output with dead-time insertion**



**Figure 297. Dead-time waveforms with delay greater than the negative pulse**



**Figure 298. 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 34.4.18: TIM1/TIM8 break and dead-time register \(TIMx\\_BDTR\)](#) 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 you to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

*Note:* When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

### 34.3.16 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM1 and TIM8 timers. 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 to enable /disable the outputs by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx\_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx\_CR2 register which are setting the output shut-down level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to [Table 218: Output control bits for complementary OCx and OCxN channels with break feature on page 1163](#) for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. You can enable the break functions by setting the BKE and BKE2 bits in the TIMx\_BDTR register. The break input polarities can be selected by configuring the BKP and BKP2 bits in the same register. BKEx and BKPx can be modified at the same time. When the BKEx and BKPx bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx\_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if you write MOE to 1 whereas it was low, you must insert a delay (dummy instruction) before reading it correctly. This is because you write the asynchronous signal and read the synchronous signal.

The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx\_OR2 and TIMx\_OR3 registers.

The sources for break (BRK) channel are:

- An external source connected to one of the BKIN pin (as per selection done in the AFIO controller), with polarity selection and optional digital filtering
- An internal source:
  - the Cortex<sup>®</sup>-M4 LOCKUP output
  - the PVD output
  - the SRAM parity error signal
  - a flash ECC error
  - a clock failure event generated by the CSS detector
  - the output from a comparator, with polarity selection and optional digital filtering
  - the analog watchdog output of the DFSDM1 peripheral

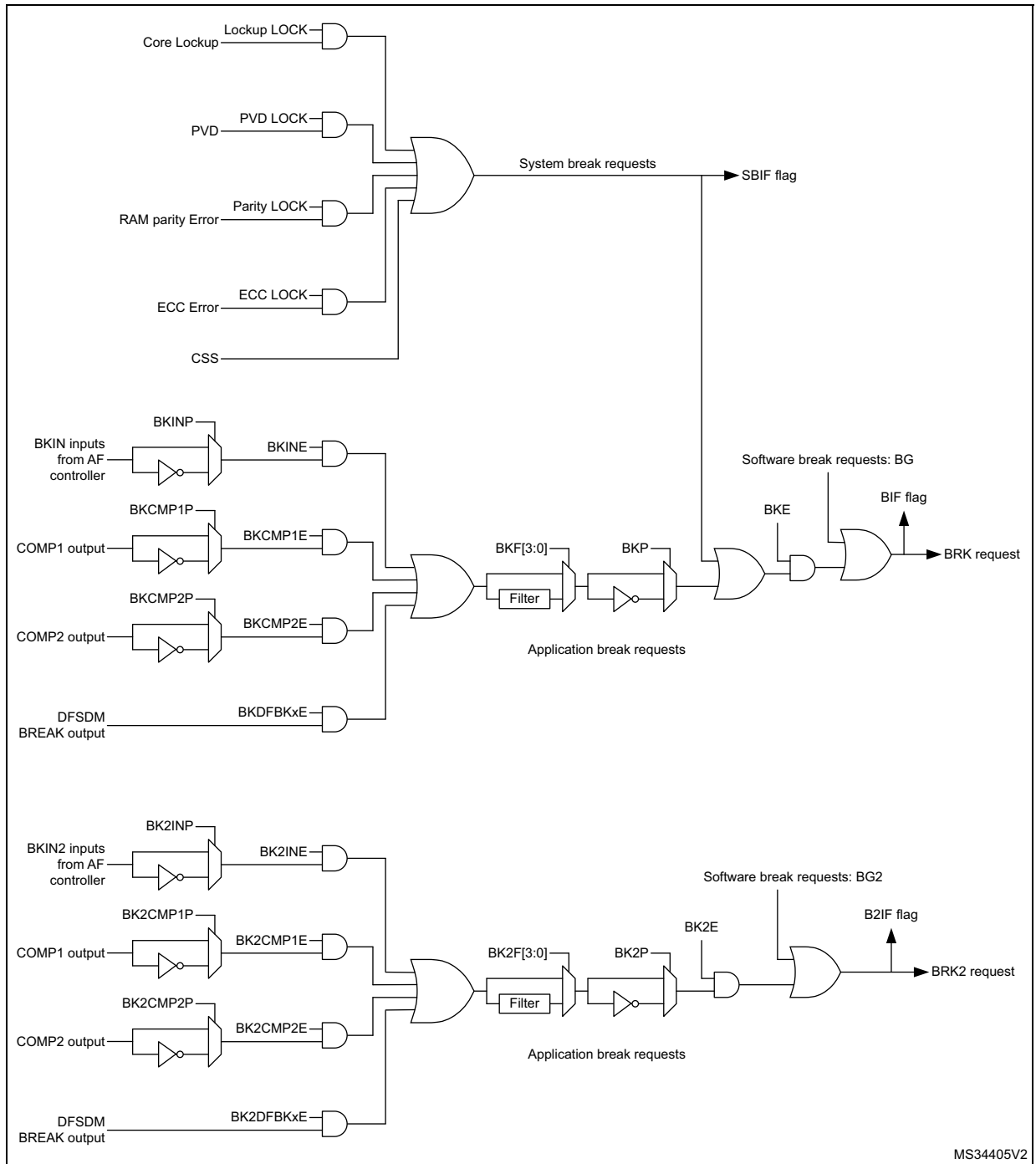
The sources for break2 (BRK2) are:

- An external source connected to one of the BKIN pin (as per selection done in the AFIO controller), 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 BKE2 enable bits values.

All sources are ORed before entering the timer BRK or BRK2 inputs, as per [Figure 299](#) below.

**Figure 299. 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. A DMA request can be sent if the BDE bit in the TIMx\_DIER register is set.
- If the AOE bit in the TIMx\_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to '1' again. In this case, it can be used for security and you can connect the break input to an alarm from power drivers, thermal sensors or any security components.

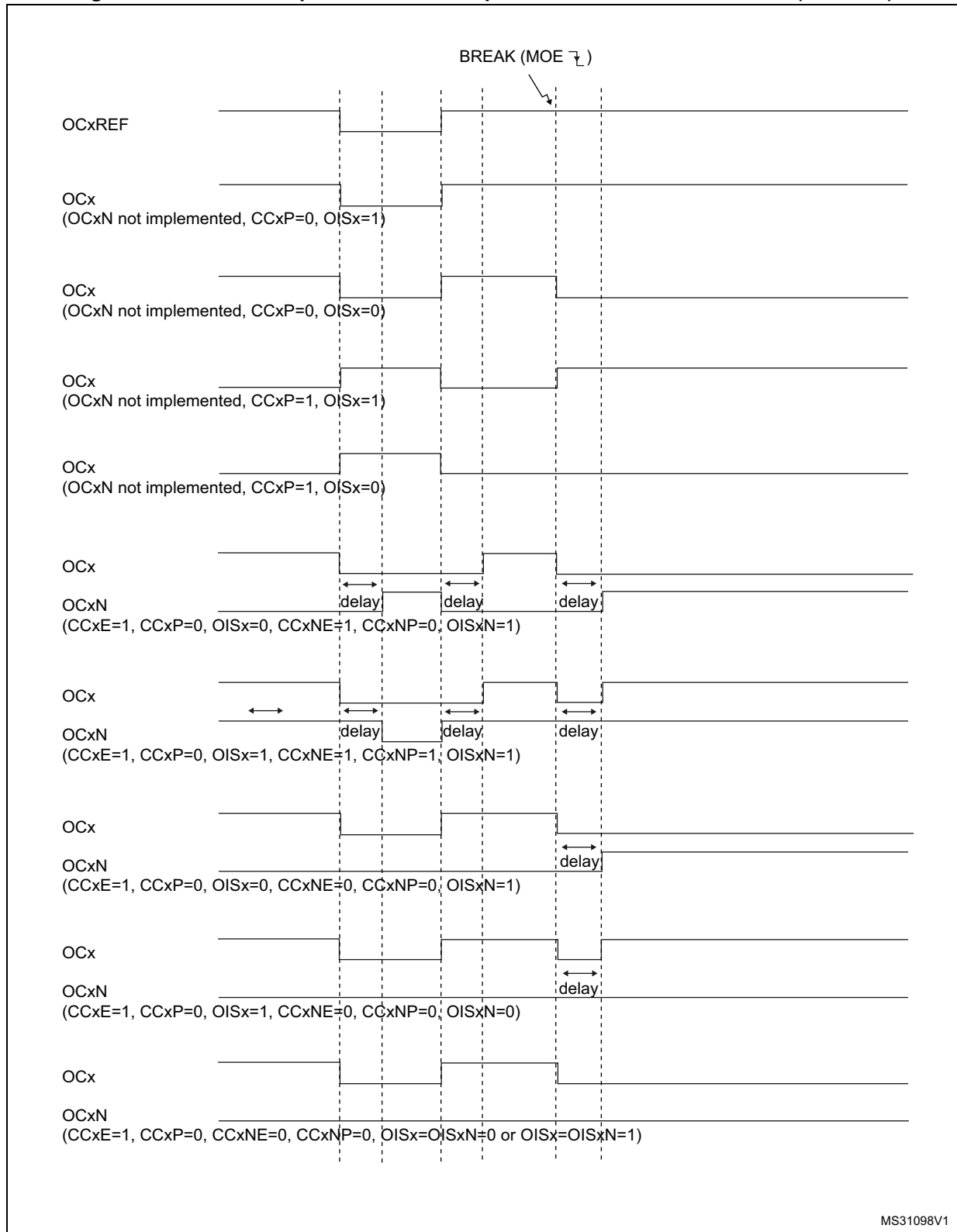
*Note:* The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows to freeze the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The application can choose from 3 levels of protection selected by the LOCK bits in the TIMx\_BDTR register. Refer to [Section 34.4.18: TIM1/TIM8 break and dead-time register \(TIMx\\_BDTR\)](#). The LOCK bits can be written only once after an MCU reset.

[Figure 300](#) shows an example of behavior of the outputs in response to a break.



Figure 300. Various output behavior in response to a break event on BRK (OSSI = 1)



The two break inputs have different behaviors on timer outputs:

- The BRK input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- BRK2 can only disable (inactive state) the PWM outputs.

The BRK has a higher priority than BRK2 input, as described in [Table 214](#).

Note: BRK2 must only be used with OSSI = OSSR = 1.

**Table 214. 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 301 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 301. PWM output state following BRK and BRK2 pins assertion (OSSI=1)**

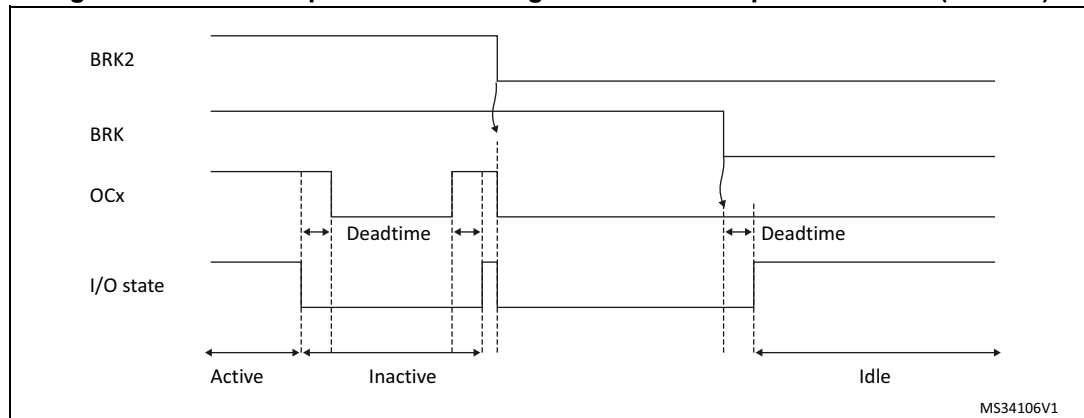
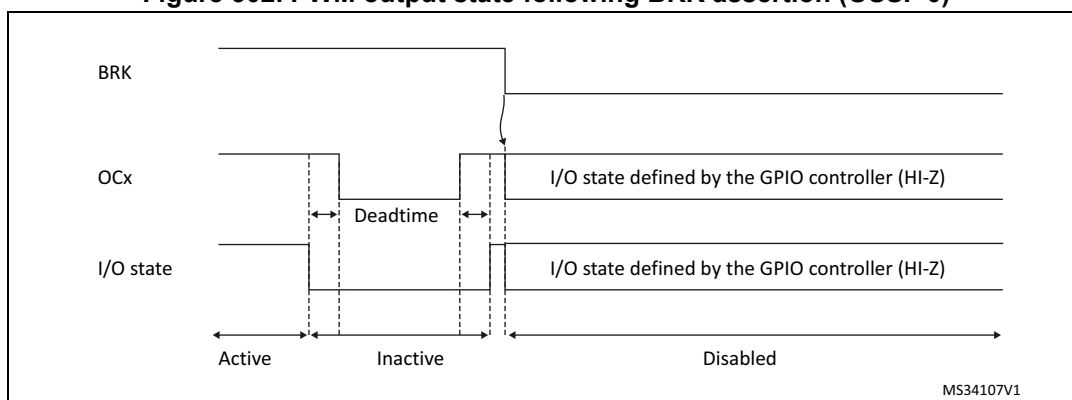


Figure 302. PWM output state following BRK assertion (OSSI=0)



### 34.3.17 Bidirectional break inputs

The TIM1/TIM8 are featuring bidirectional break I/Os, as represented on [Figure 303](#).

They allow to have:

- 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 will be active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This will prevent 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 215](#))

Table 215. 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**

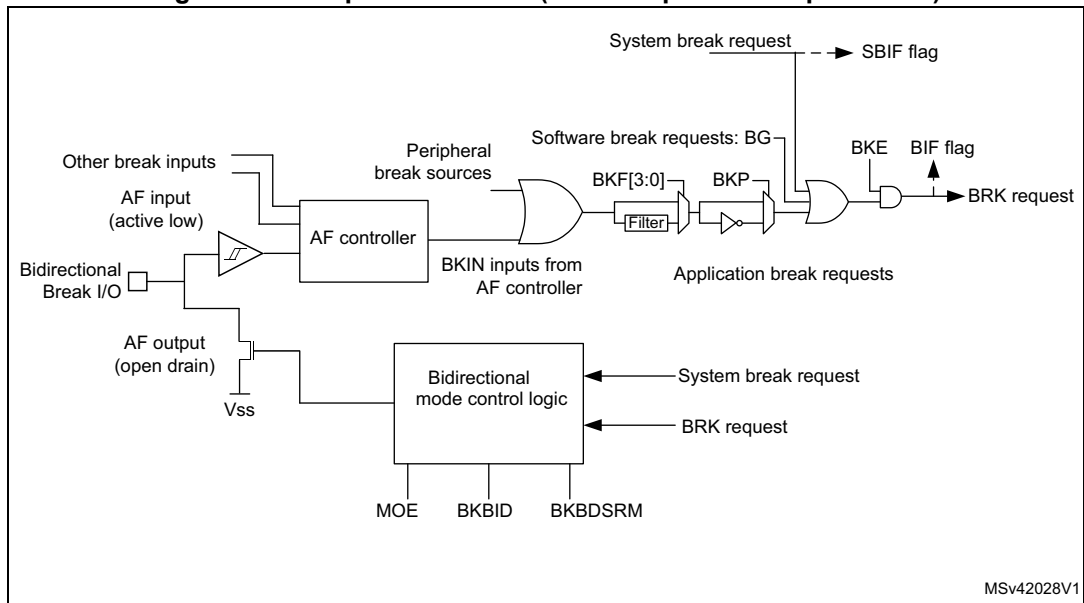
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

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 303. Output redirection (BRK2 request not represented)



**34.3.18 Clearing the OCxREF signal on an external event**

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

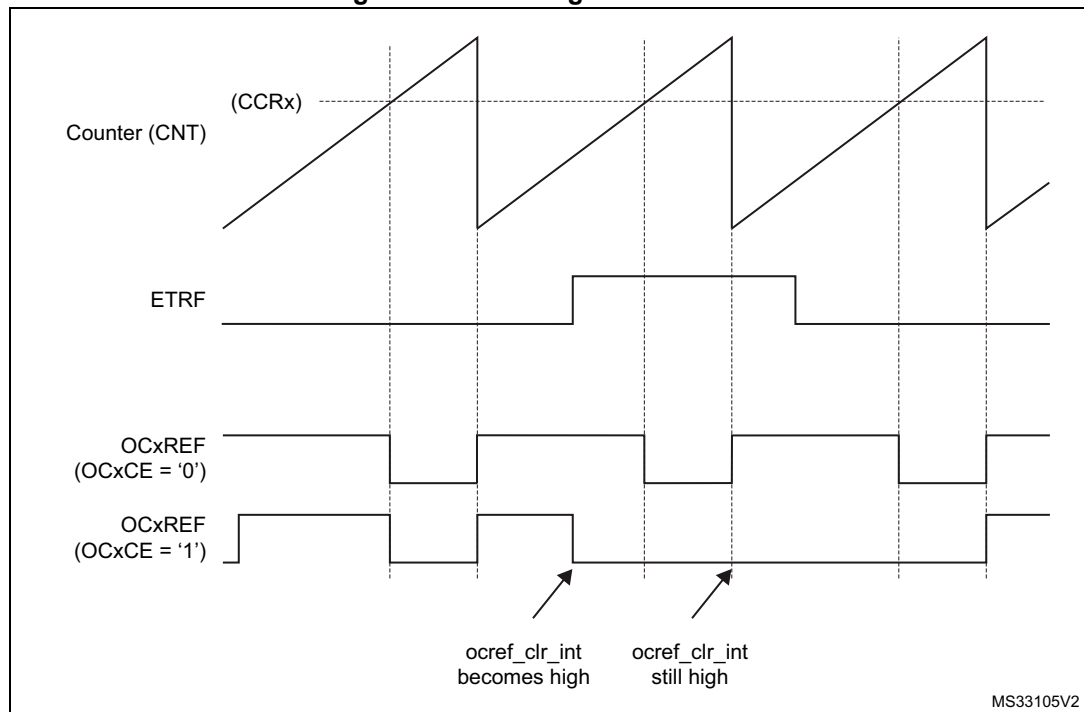
output compare and PWM modes. It does not work in forced mode. For example, the OCxREF signal can be connected to the output of a comparator to be used for current handling.

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 304 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 304. 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.

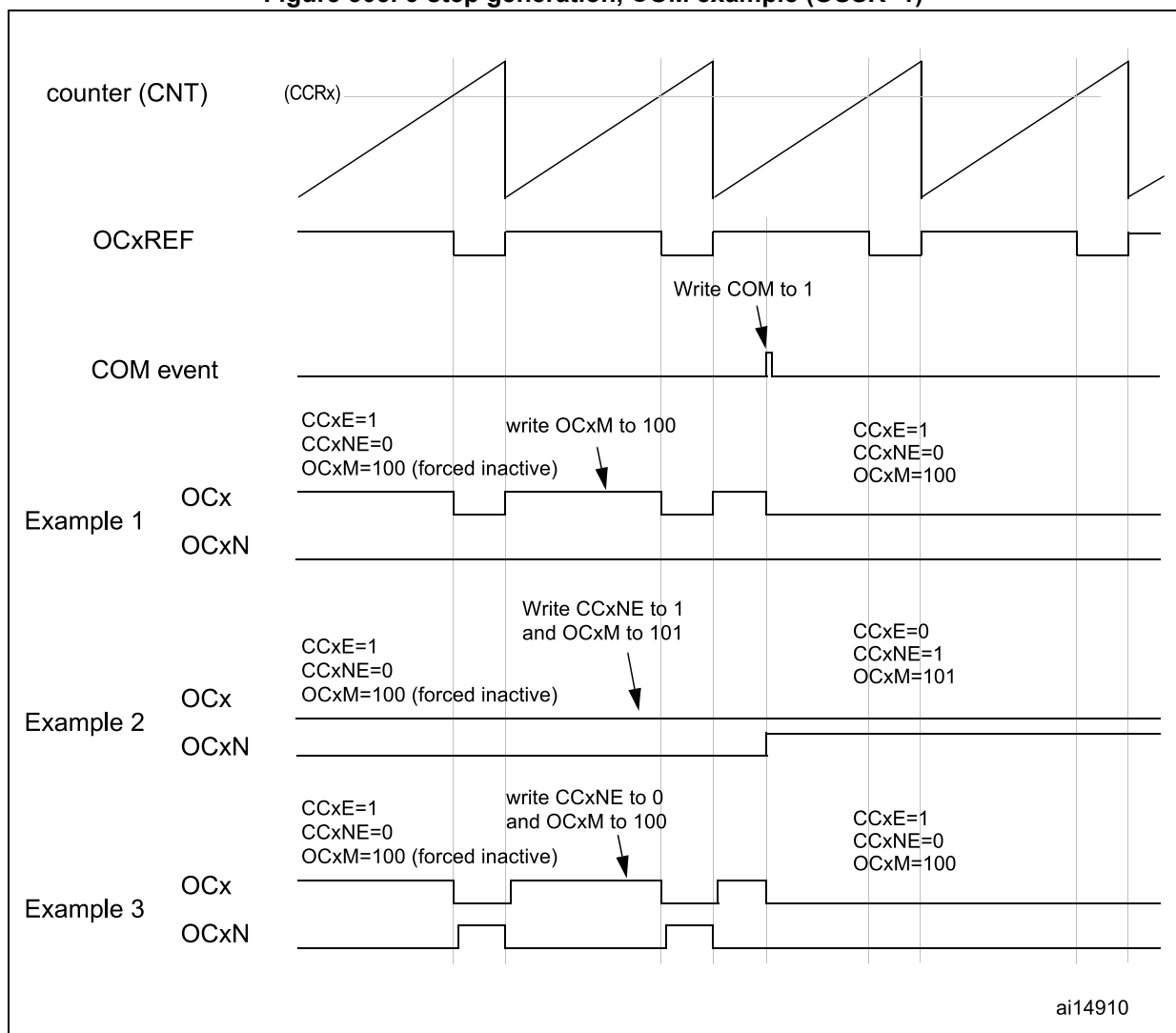
### 34.3.19 6-step PWM generation

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 you can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx\_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx\_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx\_DIER register) or a DMA request (if the COMDE bit is set in the TIMx\_DIER register).

The [Figure 305](#) describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

**Figure 305. 6-step generation, COM example (OSSR=1)**



### 34.3.20 One-pulse mode

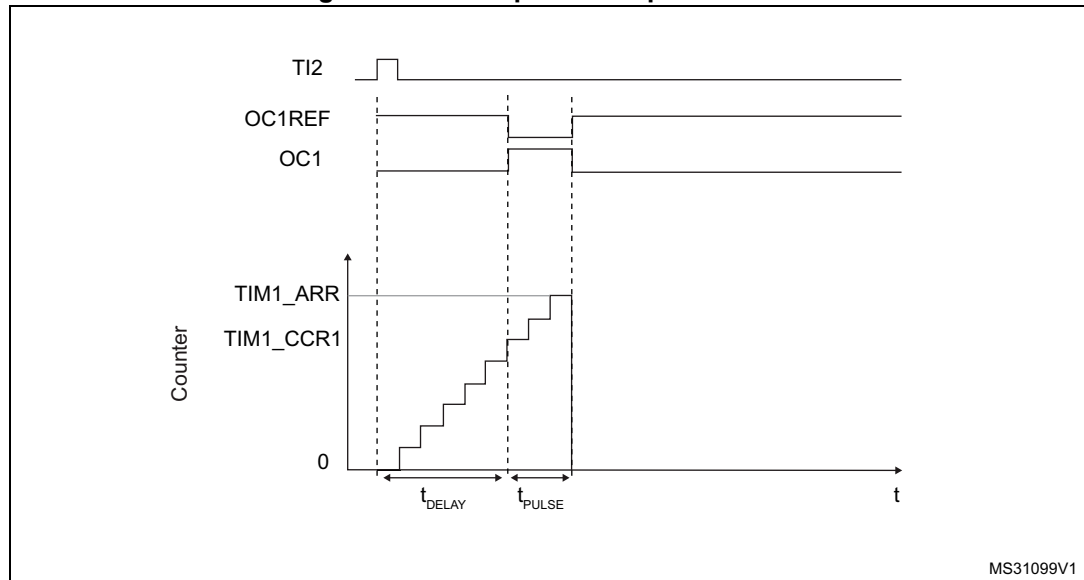
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx\_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting:  $CNT < CCRx \leq ARR$  (in particular,  $0 < CCRx$ )
- In downcounting:  $CNT > CCRx$

**Figure 306. Example of one pulse mode.**



For example you may want to generate a positive pulse on OC1 with a length of  $t_{PULSE}$  and after a delay of  $t_{DELAY}$  as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Map TI2FP2 to TI2 by writing  $CC2S='01'$  in the TIMx\_CCMR1 register.
2. TI2FP2 must detect a rising edge, write  $CC2P='0'$  and  $CC2NP='0'$  in the TIMx\_CCER register.
3. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing  $TS=110$  in the TIMx\_SMCR register.
4. TI2FP2 is used to start the counter by writing  $SMS$  to '110' in the TIMx\_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The  $t_{\text{DELAY}}$  is defined by the value written in the TIMx\_CCR1 register.
- The  $t_{\text{PULSE}}$  is defined by the difference between the auto-reload value and the compare value (TIMx\_ARR - TIMx\_CCR1).
- Let's say you want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M=111 in the TIMx\_CCMR1 register. You can optionally enable the preload registers by writing OC1PE='1' in the TIMx\_CCMR1 register and ARPE in the TIMx\_CR1 register. In this case you have to write the compare value in the TIMx\_CCR1 register, the auto-reload value in the TIMx\_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

In our example, the DIR and CMS bits in the TIMx\_CR1 register should be low.

You only want 1 pulse (Single mode), so you write '1' in the OPM bit in the TIMx\_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx\_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on Tlx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay  $t_{\text{DELAY min}}$  we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx\_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 34.3.21 Retriggerable one pulse mode (OPM)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in [Section 34.3.20](#):

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx\_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retriggerable OPM mode 1 or 2.

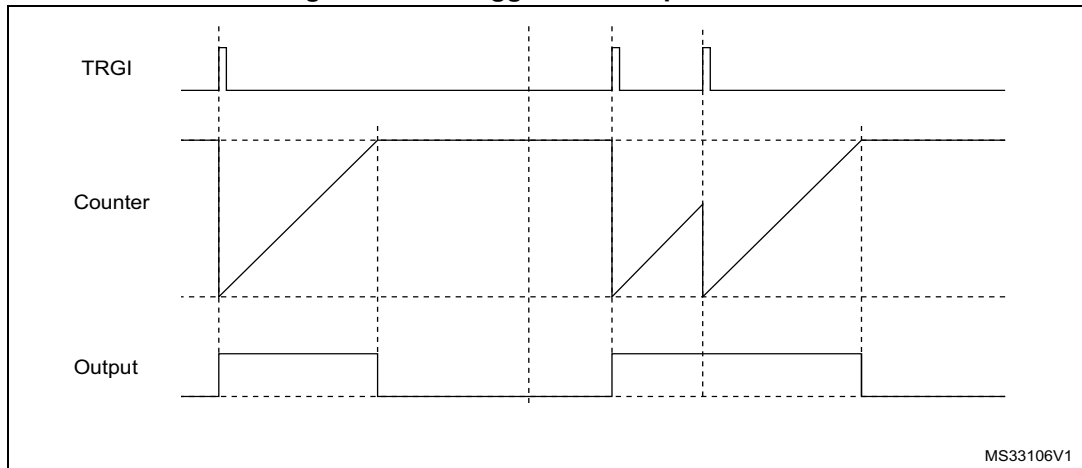
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

*Note:* The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

*This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx\_CR1.*



Figure 307. Retriggerable one pulse mode



MS33106V1

### 34.3.22 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, you can program the input filter as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to a quadrature encoder. Refer to [Table 216](#). The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in  $TIMx\_CR1$  register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the  $TIMx\_CR1$  register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the  $TIMx\_ARR$  register (0 to ARR or ARR down to 0 depending on the direction). So you must configure  $TIMx\_ARR$  before starting. In the same way, the capture, compare, 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 don't switch at the same time.

**Table 216. 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 308* gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. For this example we assume that the configuration is the following:

- CC1S='01' (TIMx\_CCMR1 register, TI1FP1 mapped on TI1).
- CC2S='01' (TIMx\_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 308. Example of counter operation in encoder interface mode.**

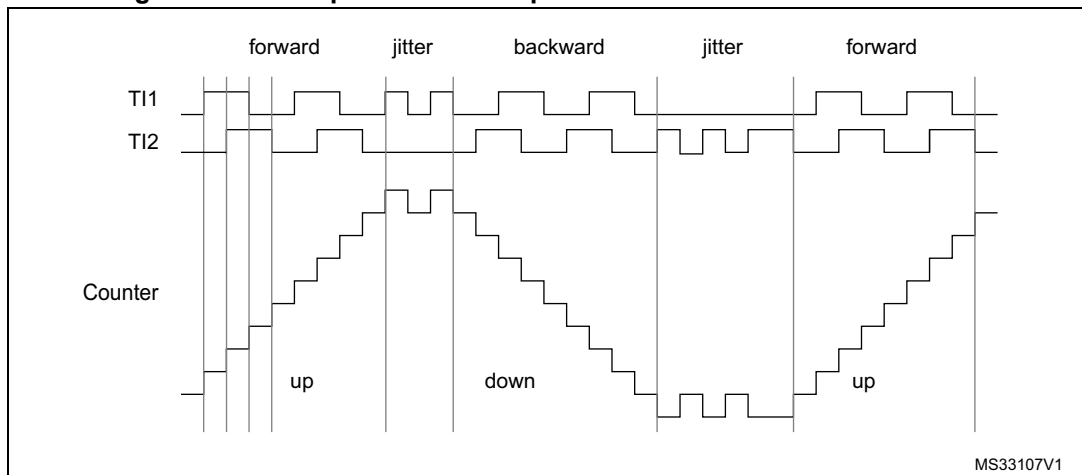
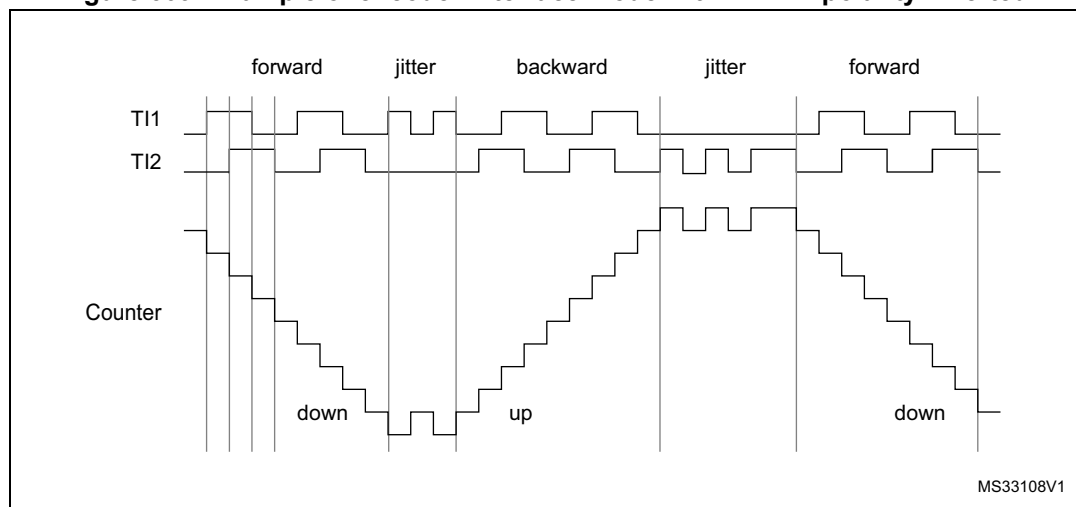


Figure 309 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

**Figure 309. Example of encoder interface mode with TI1FP1 polarity inverted.**



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. You can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. You can do this by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx\_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

### 34.3.23 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).

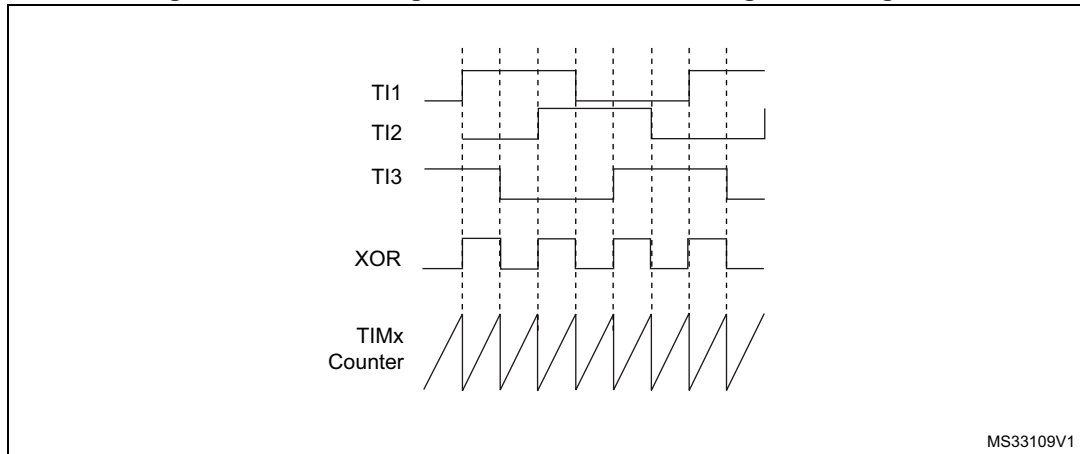
There is no latency between the UIF and UIFCPY flags assertion.

### 34.3.24 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 310](#) below.

**Figure 310. Measuring time interval between edges on 3 signals**



### 34.3.25 Interfacing with Hall sensors

This is done using the advanced-control timers (TIM1 or TIM8) to generate PWM signals to drive the motor and another timer TIMx (TIM2, TIM3, TIM4) referred to as “interfacing timer” in [Figure 311](#). 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 284: Capture/compare channel \(example: channel 1 input stage\) on page 1105](#)). The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (TIM1 or TIM8) (by triggering a COM event). The TIM1 timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer (TIM1 or TIM8) through the TRGO output.

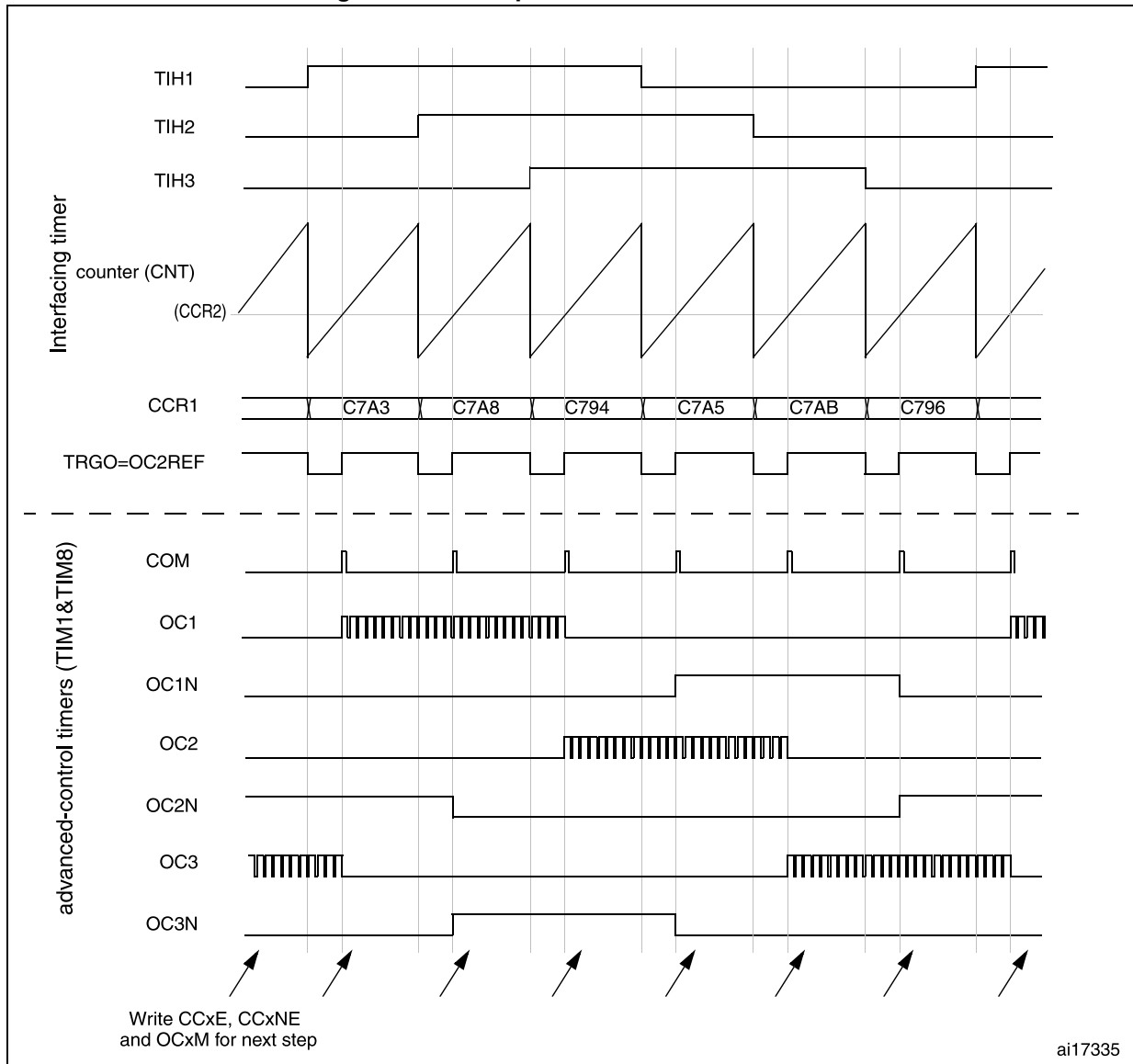
Example: you want to change the PWM configuration of your advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx\_CR2 register to '1',
- Program the time base: write the TIMx\_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program the channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx\_CCMR1 register to '01'. You can also program the digital filter if needed,
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to '111' and the CC2S bits to '00' in the TIMx\_CCMR1 register,
- Select OC2REF as trigger output on TRGO: write the MMS bits in the TIMx\_CR2 register to '101',

In the advanced-control timer TIM1, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx\_CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the TIMx\_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of OC2REF).

The [Figure 311](#) describes this example.

Figure 311. Example of Hall sensor interface



ai17335

### 34.3.26 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. They can be synchronized in several modes: Reset mode, Gated mode, and Trigger mode.

#### Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx\_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx\_ARR, TIMx\_CCRx) are updated.

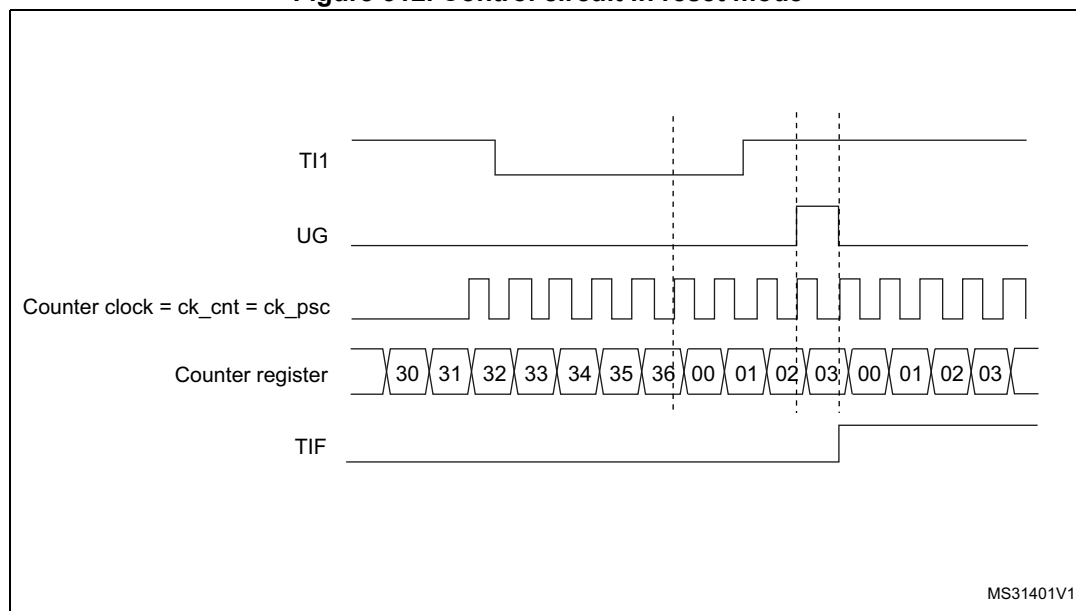
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx\_CCMR1 register. Write CC1P=0 and CC1NP='0' in TIMx\_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx\_SMCR register.
- Start the counter by writing CEN=1 in the TIMx\_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx\_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx\_DIER register).

The following figure shows this behavior when the auto-reload register TIMx\_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 312. Control circuit in reset mode



**Slave mode: Gated mode**

The counter can be enabled depending on the level of a selected input.

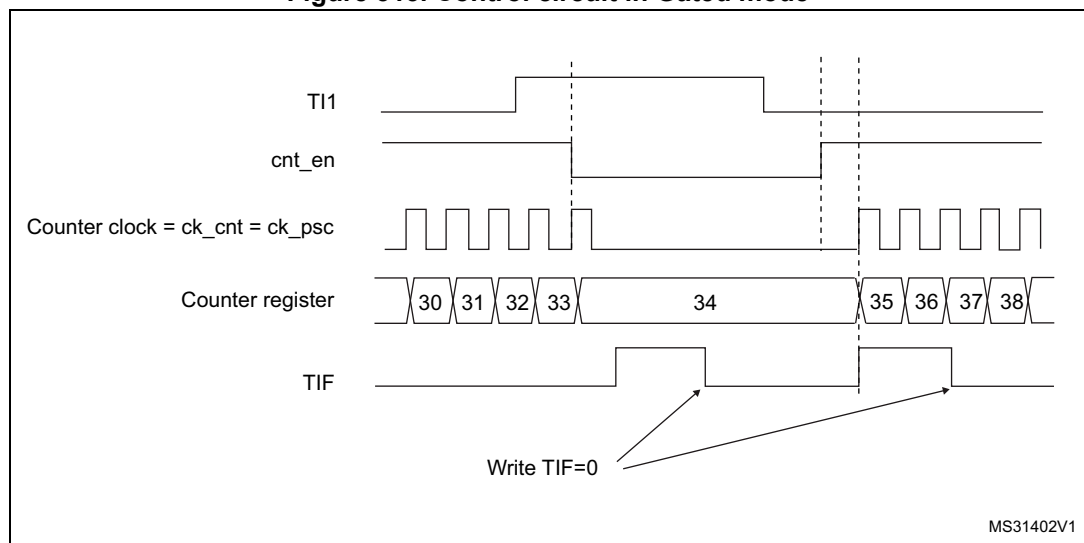
In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S=01 in TIMx\_CCMR1 register. Write CC1P=1 and CC1NP='0' in TIMx\_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx\_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx\_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx\_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

**Figure 313. Control circuit in Gated mode**



**Slave mode: Trigger mode**

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx\_CCMR1 register.



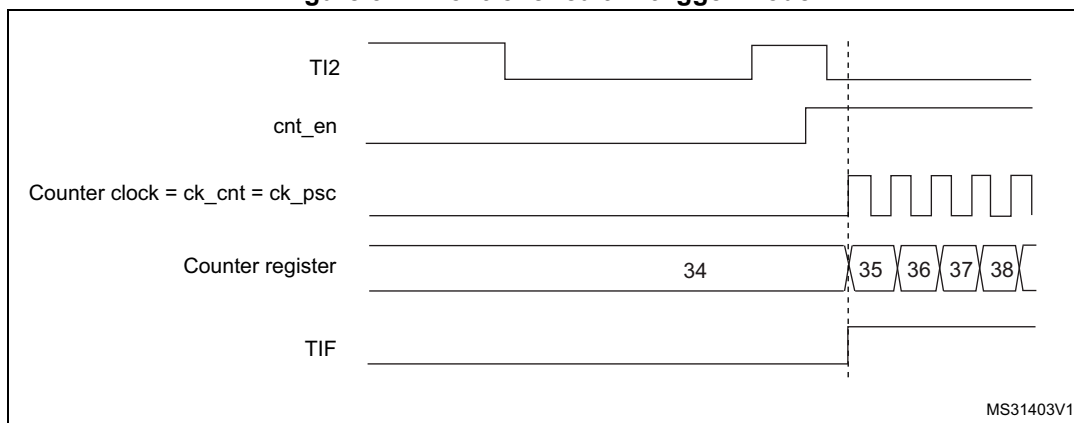
Write CC2P=1 and CC2NP=0 in TIMx\_CCER register to validate the polarity (and detect low level only).

- Configure the timer in trigger mode by writing SMS=110 in TIMx\_SMCR register. Select TI2 as the input source by writing TS=110 in TIMx\_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

**Figure 314. Control circuit in trigger mode**



**Slave mode: Combined reset + trigger mode**

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

**Slave mode: external clock mode 2 + trigger mode**

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx\_SMCR register.

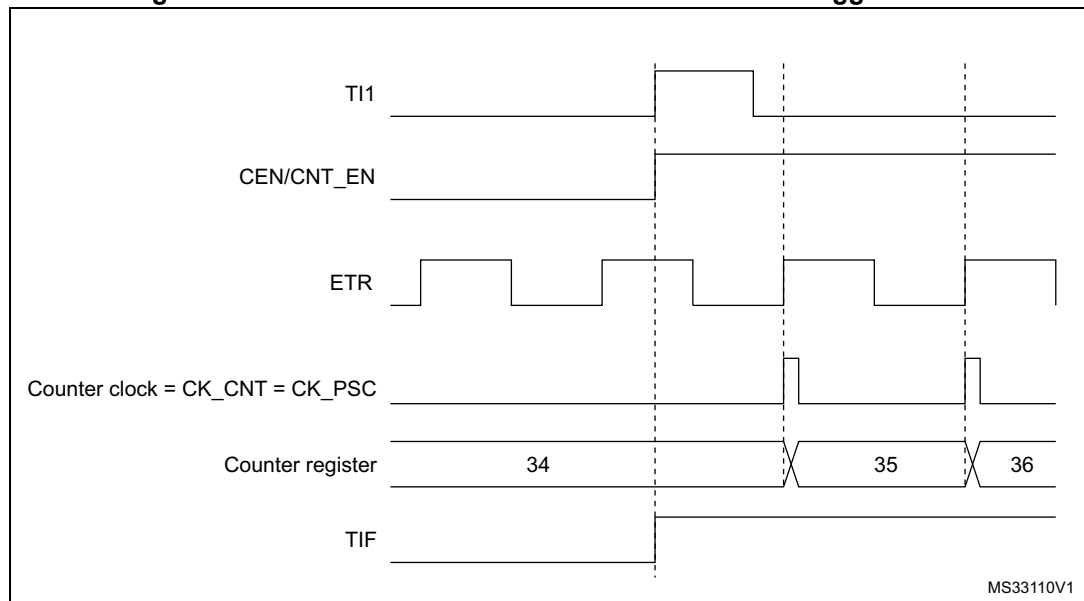
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx\_SMCR register as follows:
  - ETF = 0000: no filter
  - ETPS=00: prescaler disabled
  - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
  - IC1F=0000: no filter.
  - The capture prescaler is not used for triggering and does not need to be configured.
  - CC1S=01 in TIMx\_CCMR1 register to select only the input capture source
  - CC1P=0 and CC1NP='0' in TIMx\_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx\_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

**Figure 315. 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.

### 34.3.27 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 295 on page 1117](#).

*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.

### 34.3.28 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.

### 34.3.29 Debug mode

When the microcontroller enters debug mode (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 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), typically to force a Hi-Z.

For more details, refer to [Section 46.16.2: Debug support for timers, RTC, watchdog, bxCAN and I2C](#).

For safety purposes, when the counter is stopped (DBG\_TIMx\_STOP = 1), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

### 34.4 TIM1/TIM8 registers

Refer to for a list of abbreviations used in register descriptions.

#### 34.4.1 TIM1/TIM8 control register 1 (TIMx\_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
				r/w		r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bit 11 **UIFREMAP**: UIF status bit remapping

- 0: No remapping. UIF status bit is not copied to TIMx\_CNT register bit 31.
- 1: Remapping enabled. UIF status bit is copied to TIMx\_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK\_INT) frequency and the dead-time and sampling clock ( $t_{DTS}$ ) used by the dead-time generators and the digital filters (ETR, Tlx),

- 00:  $t_{DTS}=t_{CK\_INT}$
- 01:  $t_{DTS}=2*t_{CK\_INT}$
- 10:  $t_{DTS}=4*t_{CK\_INT}$
- 11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable

- 0: TIMx\_ARR register is not buffered
- 1: TIMx\_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

- 00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
- 01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx\_CCMRx register) are set only when the counter is counting down.
- 10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx\_CCMRx register) are set only when the counter is counting up.
- 11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx\_CCMRx register) are set both when the counter is counting up or down.

*Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)*

Bit 4 **DIR**: Direction

- 0: Counter used as upcounter
- 1: Counter used as downcounter

*Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.*

Bit 3 **OPM**: One pulse mode

- 0: Counter is not stopped at update event
- 1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

- This bit is set and cleared by software to select the UEV event sources.
- 0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
- Counter overflow/underflow
  - Setting the UG bit
  - Update generation through the slave mode controller
- 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

- This bit is set and cleared by software to enable/disable UEV event generation.
- 0: UEV enabled. The Update (UEV) event is generated by one of the following events:
- Counter overflow/underflow
  - Setting the UG bit
  - Update generation through the slave mode controller
- Buffered registers are then loaded with their preload values.
- 1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

- 0: Counter disabled
- 1: Counter enabled

*Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*

### 34.4.2 TIM1/TIM8 control register 2 (TIMx\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5
								rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	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 OR between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx\_SMCR register).

0010: **Update** - the update event is selected as trigger output (TRGO2). For instance, a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (TRGO2).

0100: **Compare** - OC1REF signal is used as trigger output (TRGO2)

0101: **Compare** - OC2REF signal is used as trigger output (TRGO2)

0110: **Compare** - OC3REF signal is used as trigger output (TRGO2)

0111: **Compare** - OC4REF signal is used as trigger output (TRGO2)

1000: **Compare** - OC5REF signal is used as trigger output (TRGO2)

1001: **Compare** - OC6REF signal is used as trigger output (TRGO2)

1010: **Compare Pulse** - OC4REF rising or falling edges generate pulses on TRGO2

1011: **Compare Pulse** - OC6REF rising or falling edges generate pulses on TRGO2

1100: **Compare Pulse** - OC4REF or OC6REF rising edges generate pulses on TRGO2

1101: **Compare Pulse** - OC4REF rising or OC6REF falling edges generate pulses on TRGO2

1110: **Compare Pulse** - OC5REF or OC6REF rising edges generate pulses on TRGO2

1111: **Compare Pulse** - OC5REF rising or OC6REF falling edges generate pulses on TRGO2

*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 to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:  
000: **Reset** - the UG bit from the TIMx\_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.  
001: **Enable** - the Counter Enable signal CNT\_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx\_SMCR register).  
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.  
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).  
100: **Compare** - OC1REF signal is used as trigger output (TRGO)  
101: **Compare** - OC2REF signal is used as trigger output (TRGO)  
110: **Compare** - OC3REF signal is used as trigger output (TRGO)  
111: **Compare** - OC4REF signal is used as trigger output (TRGO)  
*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.*

### 34.4.3 TIM1/TIM8 slave mode control register (TIMx\_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMS[3]
															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]			Res.	SMS[2:0]		
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 **SMS[3]**: Slave mode selection - bit 3  
 Refer to SMS description - bits 2:0.

Bit 15 **ETP**: External trigger polarity  
 This bit selects whether ETR or  $\overline{ETR}$  is used for trigger operations  
 0: ETR is non-inverted, active at high level or rising edge.  
 1: ETR is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable  
 This bit enables External clock mode 2.  
 0: External clock mode 2 disabled  
 1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

*Note: 1: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).  
 2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).  
 3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.*

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

- 00: Prescaler OFF
- 01: ETRP frequency divided by 2
- 10: ETRP frequency divided by 4
- 11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N 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 6:4 **TS[2:0]**: Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

- 000: Internal Trigger 0 (ITR0)
- 001: Internal Trigger 1 (ITR1)
- 010: Internal Trigger 2 (ITR2)
- 011: Internal Trigger 3 (ITR3)
- 100: TI1 Edge Detector (TI1F\_ED)
- 101: Filtered Timer Input 1 (TI1FP1)
- 110: Filtered Timer Input 2 (TI2FP2)
- 111: External Trigger input (ETRF)

See [Table 217: TIMx internal trigger connection on page 1149](#) 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.

*Note:* The other bit is at position 16 in the same register

## Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (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=100). 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 217. TIMx internal trigger connection**

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM1	TIM15	TIM2	TIM3	TIM4
TIM8	TIM1	TIM2	TIM4	TIM5

### 34.4.4 TIM1/TIM8 DMA/interrupt enable register (TIMx\_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

- Bit 15 Reserved, must be kept at reset value.
- Bit 14 **TDE**: Trigger DMA request enable  
0: Trigger DMA request disabled  
1: Trigger DMA request enabled
- Bit 13 **COMDE**: COM DMA request enable  
0: COM DMA request disabled  
1: COM DMA request enabled
- Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable  
0: CC4 DMA request disabled  
1: CC4 DMA request enabled
- Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable  
0: CC3 DMA request disabled  
1: CC3 DMA request enabled
- Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable  
0: CC2 DMA request disabled  
1: CC2 DMA request enabled
- Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable  
0: CC1 DMA request disabled  
1: CC1 DMA request enabled
- Bit 8 **UDE**: Update DMA request enable  
0: Update DMA request disabled  
1: Update DMA request enabled
- Bit 7 **BIE**: Break interrupt enable  
0: Break interrupt disabled  
1: Break interrupt enabled
- Bit 6 **TIE**: Trigger interrupt enable  
0: Trigger interrupt disabled  
1: Trigger interrupt enabled
- Bit 5 **COMIE**: COM interrupt enable  
0: COM interrupt disabled  
1: COM interrupt enabled
- Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable  
0: CC4 interrupt disabled  
1: CC4 interrupt enabled
- Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable  
0: CC3 interrupt disabled  
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable

- 0: CC2 interrupt disabled
- 1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

- 0: CC1 interrupt disabled
- 1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

- 0: Update interrupt disabled
- 1: Update interrupt enabled

### 34.4.5 TIM1/TIM8 status register (TIMx\_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6IF	CC5IF
														rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	B1F	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CC6IF**: Compare 6 interrupt flag

Refer to CC1IF description (Note: Channel 6 can only be configured as output)

Bit 16 **CC5IF**: Compare 5 interrupt flag

Refer to CC1IF description (Note: Channel 5 can only be configured as output)

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SBIF**: System Break interrupt flag

This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.

This flag must be reset to re-start PWM operation.

0: No break event occurred.

1: An active level has been detected on the system break input. An interrupt is generated if BIE=1 in the TIMx\_DIER register.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag

Refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag

Refer to CC1OF description

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag

Refer to CC1OF description

- Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag  
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.  
0: No overcapture has been detected.  
1: The counter value has been captured in TIMx\_CCR1 register while CC1IF flag was already set
- Bit 8 **B2IF**: Break 2 interrupt flag  
This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.  
0: No break event occurred.  
1: An active level has been detected on the break 2 input. An interrupt is generated if BIE=1 in the TIMx\_DIER register.
- Bit 7 **BIF**: Break interrupt flag  
This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.  
0: No break event occurred.  
1: An active level has been detected on the break input. An interrupt is generated if BIE=1 in the TIMx\_DIER register.
- Bit 6 **TIF**: Trigger interrupt flag  
This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.  
0: No trigger event occurred.  
1: Trigger interrupt pending.
- Bit 5 **COMIF**: COM interrupt flag  
This flag is set by hardware on COM event (when Capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.  
0: No COM event occurred.  
1: COM interrupt pending.
- Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag  
Refer to CC1IF description
- Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag  
Refer to CC1IF description
- Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag  
Refer to CC1IF description
- Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag  
**If channel CC1 is configured as output:** This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx\_CR1 register description). It is cleared by software.  
0: No match.  
1: The content of the counter TIMx\_CNT matches the content of the TIMx\_CCR1 register. When the contents of TIMx\_CCR1 are greater than the contents of TIMx\_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode)  
**If channel CC1 is configured as input:** This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx\_CCR1 register.  
0: No input capture occurred  
1: The counter value has been captured in TIMx\_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx\_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx\_EGR register, if URS=0 and UDIS=0 in the TIMx\_CR1 register.
- When CNT is reinitialized by a trigger event (refer to [Section 34.4.3: TIM1/TIM8 slave mode control register \(TIMx\\_SMCR\)](#)), if URS=0 and UDIS=0 in the TIMx\_CR1 register.

### 34.4.6 TIM1/TIM8 event generation register (TIMx\_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							w	w	w	w	w	w	w	w	w

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **B2G**: Break 2 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx\_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware

0: No action

1: When CCPC bit is set, it allows to update CCxE, CCxNE and OCxM bits

*Note: This bit acts only on channels having a complementary output.*

Bit 4 **CC4G**: Capture/Compare 4 generation

Refer to CC1G description

Bit 3 **CC3G**: Capture/Compare 3 generation

Refer to CC1G description

Bit 2 **CC2G**: Capture/Compare 2 generation  
 Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation  
 This bit is set by software in order to generate an event, it is automatically cleared by hardware.  
 0: No action  
 1: A capture/compare event is generated on channel 1:  
**If channel CC1 is configured as output:**  
 CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.  
**If channel CC1 is configured as input:**  
 The current value of the counter is captured in TIMx\_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation  
 This bit can be set by software, it is automatically cleared by hardware.  
 0: No action  
 1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx\_ARR) if DIR=1 (downcounting).

### 34.4.7 TIM1/TIM8 capture/compare mode register 1 (TIMx\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
Res.	Res.	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
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw



## Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **OC2M[3]**: Output Compare 2 mode - bit 3  
Refer to OC2M description on bits 14:12.

Bits 23:17 Reserved, must be kept at reset value.

Bits 16 **OC1M[3]**: Output Compare 1 mode - bit 3  
Refer to OC1M description on bits 6:4

Bit 15 **OC2CE**: Output Compare 2 clear enable

Bits 14:12 **OC2M[2:0]**: Output Compare 2 mode

Bit 11 **OC2PE**: Output Compare 2 preload enable

Bit 10 **OC2FE**: Output Compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx\_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx\_CCER).*

Bit 7 **OC1CE**: Output Compare 1 clear enable

0: OC1Ref is not affected by the ETRF input

1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 6:4 **OC1M**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx\_CCR1 and the counter TIMx\_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0011: Toggle - OC1REF toggles when TIMx\_CNT=TIMx\_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx\_CNT<TIMx\_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0') as long as TIMx\_CNT>TIMx\_CCR1 else active (OC1REF='1').

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx\_CNT<TIMx\_CCR1 else active. In downcounting, channel 1 is active as long as TIMx\_CNT>TIMx\_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

*Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S='00' (the channel is configured in output).*

*Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.*

*Note: On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.*

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx\_CCR1 disabled. TIMx\_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx\_CCR1 enabled. Read/Write operations access the preload register. TIMx\_CCR1 preload value is loaded in the active register at each update event.

*Note: 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S='00' (the channel is configured in output).*

*2: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx\_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx\_CCER).*

## Input capture mode

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

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx\_CCER).*

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N 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**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx\_CCER register).

- 00: no prescaler, capture is done each time an edge is detected on the capture input
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, IC1 is mapped on TI1
- 10: CC1 channel is configured as input, IC1 is mapped on TI2
- 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx\_CCER).

### 34.4.8 TIM1/TIM8 capture/compare mode register 2 (TIMx\_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

Refer to the above CCMR1 register description.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]
Res.	Res.	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
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE.	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

## Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **OC4M[3]**: Output Compare 4 mode - bit 3

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **OC3M[3]**: Output Compare 3 mode - bit 3

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 14:12 **OC4M**: Output compare 4 mode

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx\_CCER).*

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 6:4 **OC3M**: Output compare 3 mode

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx\_CCER).*

## Input capture mode

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

Bits 15:12 **IC4F**: Input capture 4 filter

Bits 11:10 **IC4PSC**: Input capture 4 prescaler

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx\_CCER).*

Bits 7:4 **IC3F**: Input capture 3 filter

Bits 3:2 **IC3PSC**: Input capture 3 prescaler

Bits 1:0 **CC3S**: Capture/compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx\_CCER).

### 34.4.9 TIM1/TIM8 capture/compare enable register (TIMx\_CCER)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rW	rW			rW	rW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rW		rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CC6P**: Capture/Compare 6 output polarity  
Refer to CC1P description

Bit 20 **CC6E**: Capture/Compare 6 output enable  
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/Compare 5 output polarity  
Refer to CC1P description

Bit 16 **CC5E**: Capture/Compare 5 output enable  
Refer to CC1E description

Bit 15 **CC4NP**: Capture/Compare 4 complementary output polarity  
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/Compare 4 output polarity  
Refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable  
Refer to CC1E description

Bit 11 **CC3NP**: Capture/Compare 3 complementary output polarity  
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/Compare 3 complementary output enable  
Refer to CC1NE description

- Bit 9 **CC3P**: Capture/Compare 3 output polarity  
Refer to CC1P description
- Bit 8 **CC3E**: Capture/Compare 3 output enable  
Refer to CC1E description
- Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity  
Refer to CC1NP description
- Bit 6 **CC2NE**: Capture/Compare 2 complementary output enable  
Refer to CC1NE description
- Bit 5 **CC2P**: Capture/Compare 2 output polarity  
Refer to CC1P description
- Bit 4 **CC2E**: Capture/Compare 2 output enable  
Refer to CC1E description
- Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity  
**CC1 channel configured as output:**  
 0: OC1N active high.  
 1: OC1N active low.  
**CC1 channel configured as input:**  
 This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.  
*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S="00" (channel configured as output).*  
*On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.*
- Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable  
 0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.  
 1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.  
*On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 1 **CC1P**: Capture/Compare 1 output polarity

**CC1 channel configured as output:**

0: OC1 active high

1: OC1 active low

**CC1 channel configured as input:** CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

00: non-inverted/rising edge. The circuit is sensitive to TlxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TlxFP1 is not inverted (trigger operation in gated mode or encoder mode).

01: inverted/falling edge. The circuit is sensitive to TlxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TlxFP1 is inverted (trigger operation in gated mode or encoder mode).

10: reserved, do not use this configuration.

11: non-inverted/both edges/ The circuit is sensitive to both TlxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TlxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

*On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 0 **CC1E**: Capture/Compare 1 output enable

**CC1 channel configured as output:**

0: Off - OC1 is not active. OC1 level is then function of MOE, OSS1, OSSR, OIS1, OIS1N and CC1NE bits.

1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSS1, OSSR, OIS1, OIS1N and CC1NE bits.

**CC1 channel configured as input:** This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx\_CCR1) or not.

0: Capture disabled.

1: Capture enabled.

*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 218. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states <sup>(1)</sup>	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	X	0	0	Output disabled (not driven by the timer: Hi-Z) OCx=0, OCxN=0	
		0	0	1	Output disabled (not driven by the timer: Hi-Z) OCx=0	OCxREF + Polarity OCxN = OCxREF xor CCxNP
		0	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP	Output Disabled (not driven by the timer: Hi-Z) OCxN=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP	OCxREF + Polarity OCxN = OCxREF x or CCxNP
		1	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP	Off-State (output enabled with inactive state) OCxN=CCxNP
0	0	X	X	X	Output disabled (not driven by the timer anymore). The output state is defined by the GPIO controller and can be High, Low or Hi-Z.	
	1		0	0	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP (if BRK or BRK2 is triggered).	
			0	1	Then (this is valid only if BRK is triggered), if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCx and OCxN both in active state (may cause a short circuit when driving switches in half-bridge configuration).	
			1	0		
			1	1	<b>Note:</b> BRK2 can only be used if OSSI = OSSR = 1.	

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

*Note:* The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIO registers.

### 34.4.10 TIM1/TIM8 counter (TIMx\_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx\_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

### 34.4.11 TIM1/TIM8 prescaler (TIMx\_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK\_CNT) is equal to  $f_{CK\_PSC} / (PSC[15:0] + 1)$ .

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx\_EGR register or through trigger controller when configured in “reset mode”).

### 34.4.12 TIM1/TIM8 auto-reload register (TIMx\_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the [Section 34.3.1: Time-base unit on page 1085](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 34.4.13 TIM1/TIM8 repetition counter register (TIMx\_RCR)

Address offset: 0x30

Reset value: 0x0000

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

### 34.4.14 TIM1/TIM8 capture/compare register 1 (TIMx\_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

**If channel CC1 is configured as output:** CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signaled on OC1 output.

**If channel CC1 is configured as input:** 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.

### 34.4.15 TIM1/TIM8 capture/compare register 2 (TIMx\_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

**If channel CC2 is configured as output:** CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx\_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.

### 34.4.16 TIM1/TIM8 capture/compare register 3 (TIMx\_CCR3)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR3[15:0]**: Capture/Compare value

**If channel CC3 is configured as output:** CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx\_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.

### 34.4.17 TIM1/TIM8 capture/compare register 4 (TIMx\_CCR4)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR4[15:0]**: Capture/Compare value

**If channel CC4 is configured as output:** CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx\_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.

### 34.4.18 TIM1/TIM8 break and dead-time register (TIMx\_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[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, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx\_BDTR register.

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

Bit 29 **BK2BID**: Break2 bidirectional  
Refer to BKBID description

- Bit 28 **BKBID**: Break Bidirectional
- 0: Break input BRK in input mode
  - 1: Break input BRK in bidirectional mode
- In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.
- Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*
- Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*
- Bit 27 **BK2DSRM**: Break2 Disarm
- Refer to BKDSRM description
- Bit 26 **BKDSRM**: Break Disarm
- 0: Break input BRK is armed
  - 1: Break input BRK is disarmed
- This bit is cleared by hardware when no break source is active.
- The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.
- Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*
- Bit 25 **BK2P**: Break 2 polarity
- 0: Break input BRK2 is active low
  - 1: Break input BRK2 is active high
- Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*
- Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*
- Bit 24 **BK2E**: Break 2 enable
- This bit enables the complete break 2 protection (including all sources connected to bk\_acth and BKIN sources, as per [Figure 299: Break and Break2 circuitry overview](#)).
  - 0: Break2 function disabled
  - 1: Break2 function enabled
- Note: The BRKIN2 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 OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx\_CCER register).

See OC/OCN enable description for more details ([Section 34.4.9: TIM1/TIM8 capture/compare enable register \(TIMx\\_CCER\)](#)).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs BRK and BRK2 is active)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to bk\_ach and BKIN sources, as per [Figure 299: 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 34.4.9: TIM1/TIM8 capture/compare enable register \(TIMx\\_CCER\)](#)).

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 34.4.9: TIM1/TIM8 capture/compare enable register \(TIMx\\_CCER\)](#)).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIMx\_BDTR register, OISx and OISxN bits in TIMx\_CR2 register and BK2BID, BKBID, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits in TIMx\_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx\_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx\_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

*Note: The LOCK bits can be written only once after the reset. Once the TIMx\_BDTR register has been written, their content is frozen until the next reset.*

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5]=0xx => DT=DTG[7:0]x t<sub>dtg</sub> with t<sub>dtg</sub>=t<sub>DTS</sub>.

DTG[7:5]=10x => DT=(64+DTG[5:0])x t<sub>dtg</sub> with T<sub>dtg</sub>=2x t<sub>DTS</sub>.

DTG[7:5]=110 => DT=(32+DTG[4:0])x t<sub>dtg</sub> with T<sub>dtg</sub>=8x t<sub>DTS</sub>.

DTG[7:5]=111 => DT=(32+DTG[4:0])x t<sub>dtg</sub> with T<sub>dtg</sub>=16x t<sub>DTS</sub>.

Example if T<sub>DTS</sub>=125ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 us to 31750 ns by 250 ns steps,

32 us to 63us by 1 us steps,

64 us to 126 us by 2 us steps

*Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

### 34.4.19 TIM1/TIM8 DMA control register (TIMx\_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

- 00000: 1 transfer
- 00001: 2 transfers
- 00010: 3 transfers
- ...
- 10001: 18 transfers

**Example:** Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2\_CR1.

– If DBL = 7 bytes and DBA = TIM2\_CR1 represents the address of the byte to be transferred, the address of the transfer should be given by the following equation:

(TIMx\_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx\_CR1 address) + DBA, which gives us the address from/to which the data will be copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx\_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

- If you configure the DMA Data Size in half-words, 16-bit data will be transferred to each of the 7 registers.
- If you configure the DMA Data Size in bytes, the data will also be transferred to 7 registers: the first register will contain the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, you also have to specify the size of data transferred by DMA.

Bits 7:5 Reserved, 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,
- ...

### 34.4.20 TIM1/TIM8 DMA address for full transfer (TIMx\_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]																
	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]																
	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **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).

### 34.4.21 TIM1 option register 1 (TIM1\_OR1)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T11_RMP	Res.	Res.	ETR_ADC1_RMP	
											rw			rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **T11\_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 **ETR\_ADC1\_RMP**: External trigger remap on ADC1 analog watchdog

00 : TIM1\_ETR is not connected to ADC1 AWDx. This configuration must be selected when the ETR comes from the I/O.

01 : TIM1\_ETR is connected to ADC1 AWD1.

10 : TIM1\_ETR is connected to ADC1 AWD2.

11 : TIM1\_ETR is connected to ADC1 AWD3.

*Note: ADC1 AWDx sources are 'ORed' with the TIM1\_ETR input signals. When ADC1 AWDx is used, it is necessary to make sure that the corresponding TIM1\_ETR input pin is not enabled in the alternate function controller. Refer to [Figure 277: TIM1 ETR input circuitry](#).*

### 34.4.22 TIM8 option register 1 (TIM8\_OR1)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T11_RMP	Res.	Res.	Res.	Res.
											rw				

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **T11\_RMP**: Input Capture 1 remap

0: TIM8 input capture 1 is connected to I/O

1: TIM8 input capture 1 is connected to COMP2 output.

Bits 3:0 Reserved, must be kept at reset value.

### 34.4.23 TIM1/TIM8 capture/compare mode register 3 (TIMx\_CCMR3)

Address offset: 0x54

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channels 5 and 6 can only be configured in output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]
							rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6 CE	OC6M[2:0]			OC6 PE	OC6FE	Res.	Res.	OC5 CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

#### Output compare mode

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

Bits 24, 14, 13, 12 **OC6M[3:0]**: Output compare 6 mode

Bit 11 **OC6PE**: Output compare 6 preload enable

Bit 10 **OC6FE**: Output compare 6 fast enable

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

Bit 7 **OC5CE**: Output compare 5 clear enable

Bits 16, 6, 5, 4 **OC5M[3:0]**: Output compare 5 mode

Bit 3 **OC5PE**: Output compare 5 preload enable

Bit 2 **OC5FE**: Output compare 5 fast enable

Bits 1:0 Reserved, must be kept at reset value.

### 34.4.24 TIM1/TIM8 capture/compare register 5 (TIMx\_CCR5)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r/w	r/w	r/w													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bit 31 **GC5C3**: Group Channel 5 and Channel 3  
 Distortion on Channel 3 output:  
 0: No effect of OC5REF on OC3REFC  
 1: OC3REFC is the logical AND of OC3REFC and OC5REF  
 This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2).  
*Note: it is also possible to apply this distortion on combined PWM signals.*

Bit 30 **GC5C2**: Group Channel 5 and Channel 2  
 Distortion on Channel 2 output:  
 0: No effect of OC5REF on OC2REFC  
 1: OC2REFC is the logical AND of OC2REFC and OC5REF  
 This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).  
*Note: it is also possible to apply this distortion on combined PWM signals.*

Bit 29 **GC5C1**: Group Channel 5 and Channel 1  
 Distortion on Channel 1 output:  
 0: No effect of OC5REF on OC1REFC5  
 1: OC1REFC is the logical AND of OC1REFC and OC5REF  
 This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).  
*Note: it is also possible to apply this distortion on combined PWM signals.*

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR5[15:0]**: Capture/Compare 5 value  
 CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs.  
 The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signaled on OC5 output.

### 34.4.25 TIM1/TIM8 capture/compare register 6 (TIMx\_CCR6)

Address offset: 0x5C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR6[15:0]**: Capture/Compare 6 value

CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 register when an update event occurs.  
The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signaled on OC6 output.

### 34.4.26 TIM1 option register 2 (TIM1\_OR2)

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.	ETR SEL [2]
															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	BKDF1 BKOE	Res.	Res.	Res.	Res.	Res.	BK CMP2E	BK CMP1E	BKINE
rw	rw			rw	rw	rw	rw						rw	rw	rw

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

Bits 16:14 **ETRSEL[2:0]**: ETR source selection

These bits select the ETR input source.  
000: ETR input is connected to I/O  
001: COMP1 output  
010: COMP2 output  
011: ADC1 AWD1  
100: ADC1 AWD2  
101: ADC1 AWD3  
Others: Reserved

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

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

- Bit 11 **BKCMP2P**: 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 high  
1: COMP2 input is active low  
*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*
- Bit 10 **BKCMP1P**: BRK COMP1 input polarity  
This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.  
0: COMP1 input is active high  
1: COMP1 input is active low  
*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 high  
1: BKIN input is active low  
*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*
- Bit 8 **BKDF1BK0E**: BRK dfsdm1\_break[0] enable  
This bit enables the dfsdm1\_break[0] for the timer's BRK input. dfsdm1\_break[0] output is 'ORed' with the other BRK sources.  
0: dfsdm1\_break[0] input disabled  
1: dfsdm1\_break[0] input enabled  
*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*
- Bits 7:3 Reserved, must be kept at reset value.
- Bit 2 **BKCMP2E**: BRK COMP2 enable  
This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.  
0: COMP2 input disabled  
1: COMP2 input enabled  
*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*
- Bit 1 **BKCMP1E**: BRK COMP1 enable  
This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.  
0: COMP1 input disabled  
1: COMP1 input enabled  
*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer’s BRK input. BKIN input is ‘ORed’ with the other BRK sources.

- 0: BKIN input disabled
- 1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Note: Refer to Figure 277: TIM1 ETR input circuitry and to Figure 299: Break and Break2 circuitry overview.*

### 34.4.27 TIM1 option register 3 (TIM1\_OR3)

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	BK2DF1 BK1E	Res.	Res.	Res.	Res.	Res.	BK2 CMP2E	BK2 CMP1E	BK2INE
				rw	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 BKP2 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 **BK2CMP1P**: BRK2 COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP2 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 **BK2INP**: BRK2 BKIN2 input polarity

This bit selects the BKIN2 alternate function input sensitivity. It must be programmed together with the BKP2 polarity bit.

- 0: BKIN2 input is active low
- 1: BKIN2 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 8 **BK2DF1BK1E**: BRK2 dfsdm1\_break[1] enable

This bit enables the dfsdm1\_break[1] for the timer's BRK2 input. dfsdm1\_break[1] output is 'ORed' with the other BRK2 sources.

- 0: dfsdm1\_break[1] input disabled
- 1: dfsdm1\_break[1] input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 7:3 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 299: Break and Break2 circuitry overview](#).*

### 34.4.28 TIM8 option register 2 (TIM8\_OR2)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [2]
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]	Res.	Res.	BK CMP2 P	BK CMP1 P	BKINP	BKDF1 BK2E	Res.	Res.	Res.	Res.	Res.	BK CMP2E	BK CMP1E	BKINE	
rw	rw		rw	rw	rw	rw						rw	rw	rw	

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



- Bits 16:14 **ETRSEL[2:0]**: ETR source selection  
These bits select the ETR input source.  
000: ETR input is connected to I/O  
001: COMP1 output  
010: 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 **BKCMP2P**: 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 high  
1: COMP2 input is active low  
*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).
- Bit 10 **BKCMP1P**: BRK COMP1 input polarity  
This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.  
0: COMP1 input is active high  
1: COMP1 input is active low  
*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 high  
1: BKIN input is active low  
*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).
- Bit 8 **BKDF1BK2E**: BRK dfsdm1\_break[2] enable  
This bit enables the dfsdm1\_break[2] for the timer's BRK input. dfsdm1\_break[2] output is 'ORed' with the other BRK sources.  
0: dfsdm1\_break[2] input disabled  
1: dfsdm1\_break[2] input enabled  
*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).
- Bits 7:3 Reserved, must be kept at reset value.
- Bit 2 **BKCMP2E**: BRK COMP2 enable  
This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.  
0: COMP2 input disabled  
1: COMP2 input enabled  
*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

Bit 1 **BKCOMP1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer’s BRK input. COMP1 output is ‘ORed’ with the other BRK sources.

- 0: COMP1 input disabled
- 1: COMP1 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer’s BRK input. BKIN input is ‘ORed’ with the other BRK sources.

- 0: BKIN input disabled
- 1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Note: Refer to Figure 278: TIM8 ETR input circuitry and to Figure 299: Break and Break2 circuitry overview.*

### 34.4.29 TIM8 option register 3 (TIM8\_OR3)

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	BK2DF1 BK3E	Res.	Res.	Res.	Res.	Res.	BK2 CMP2E	BK2 CMP1E	BK2INE
				rw	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 BKP2 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 **BK2CMP1P**: BRK2 COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP2 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 BK2INP:** BRK2 BKIN2 input polarity

This bit selects the BKIN2 alternate function input sensitivity. It must be programmed together with the BKP2 polarity bit.

0: BKIN2 input is active low

1: BKIN2 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 8 BK2DF1BK3E:** BRK2 dfsdm1\_break[3] enable

This bit enables the dfsdm1\_break[3] for the timer's BRK2 input. dfsdm1\_break[3] output is 'ORed' with the other BRK2 sources.

0: dfsdm1\_break[3] input disabled

1: dfsdm1\_break[3] input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 7:3 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 299: Break and Break2 circuitry overview](#).*

### 34.4.30 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

**Table 219. 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.	UIS	Res.	CKD [1:0]	ARPE	Res.	CMS [1:0]	DIR	OPM	URS	UDIS	CEN		
	Reset value																						0		0	0	0	0	0	0	0	0	0	
0x04	TIM1_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIS2N	UIS2	UIS1N	UIS1	UIS	Res.	MMS [2:0]	CCDS	CCUS	Res.	CCPC		
	Reset value										0	0	0	0		0		0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x08	TIM1_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	
	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.	
	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.	
	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.	
	Reset value																																	
	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.	
Reset value																																		
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.	
	Reset value																																	
	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.	
Reset value																																		
0x20	TIM1_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																	
0x24	TIM1_CNT	UIFCP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value	0																																





Table 219. TIM1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5C	TIM1_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x60	TIM1_OR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [2:0]		Res.	Res.	BK2CMP2P	BK2CMP1P	BK2INP	BK2DF1BK0E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	0			0	0	0	0						0	0
0x64	TIM1_OR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2CMP2P	BK2CMP1P	BK2INP	BK2DF1BK1E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																					0	0	0	0							0	0

Refer to [Section 2.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

### 34.4.31 TIM8 register map

TIM8 registers are mapped as 16-bit addressable registers as described in the table below:

**Table 220. TIM8 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	TIM8_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]	ARPE	Res.	CMS [1:0]	DIR	OPM	URS	UDIS	CEN		
	Reset value																						0		0	0	0	0	0	0	0	0	0	
0x04	TIM8_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]	OIS6	OIS5	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TIS	Res.	MMS [2:0]	CCDS	CCUS	Res.	CCFC	
	Reset value										0	0	0	0			0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x08	TIM8_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMS[3]	ETP	ECE	ETP [1:0]	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	
0x0C	TIM8_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x10	TIM8_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.	
	Reset value																0	0																
0x14	TIM8_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.	
	Reset value																																	
0x18	TIM8_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.	
	Reset value																																	
	TIM8_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.	
Reset value																																		
0x1C	TIM8_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.	
	Reset value																																	
	TIM8_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.	
Reset value																																		
0x20	TIM8_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																	
0x24	TIM8_CNT	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value	0																																





Table 220. TIM8 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	TIM8_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIM8_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	TIM8_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIM8_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	TIM8_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TIM8_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	TIM8_BDTR	Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]			BKF[3:0]			MOE	AOE	BKP	BKE	OSSR	OSSI	LOK [1:0]	DT[7:0]										
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TIM8_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]				Res.	Res.	Res.	DBA[4:0]						
	Reset value																				0	0	0	0	0					0	0	0	0
0x4C	TIM8_DMAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	TIM8_CCMR3 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]	OC6CE	OC6M [2:0]		OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]		OC5PE	OC5FE	Res.	Res.	
	Reset value								0									0	0	0	0	0	0			0	0	0	0	0	0	0	
0x58	TIM8_CCR5	GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[15:0]															
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 220. TIM8 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	
0x5C	TIM8_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x60	TIM8_OR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [2:0]		Res.	Res.	Res.	BKCOMP2P	BKCOMP1P	BKINP	BKDF1BK2E	Res.	Res.	Res.	Res.	Res.	Res.	BKCOMP2E	BKCOMP1E	BKINE
	Reset value																	0	0	0			0	0	0	0						0	0	1
0x64	TIM8_OR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2CMP2P	BK2CMP1P	BK2INP	BK2DF1BK3E	Res.	Res.	Res.	Res.	Res.	Res.	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value																						0	0	0	0						0	0	1

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.

## 35 General-purpose timers (TIM2/TIM3/TIM4/TIM5)

### 35.1 TIM2/TIM3/TIM4/TIM5 introduction

The general-purpose timers consist of a 16-bit or 32-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

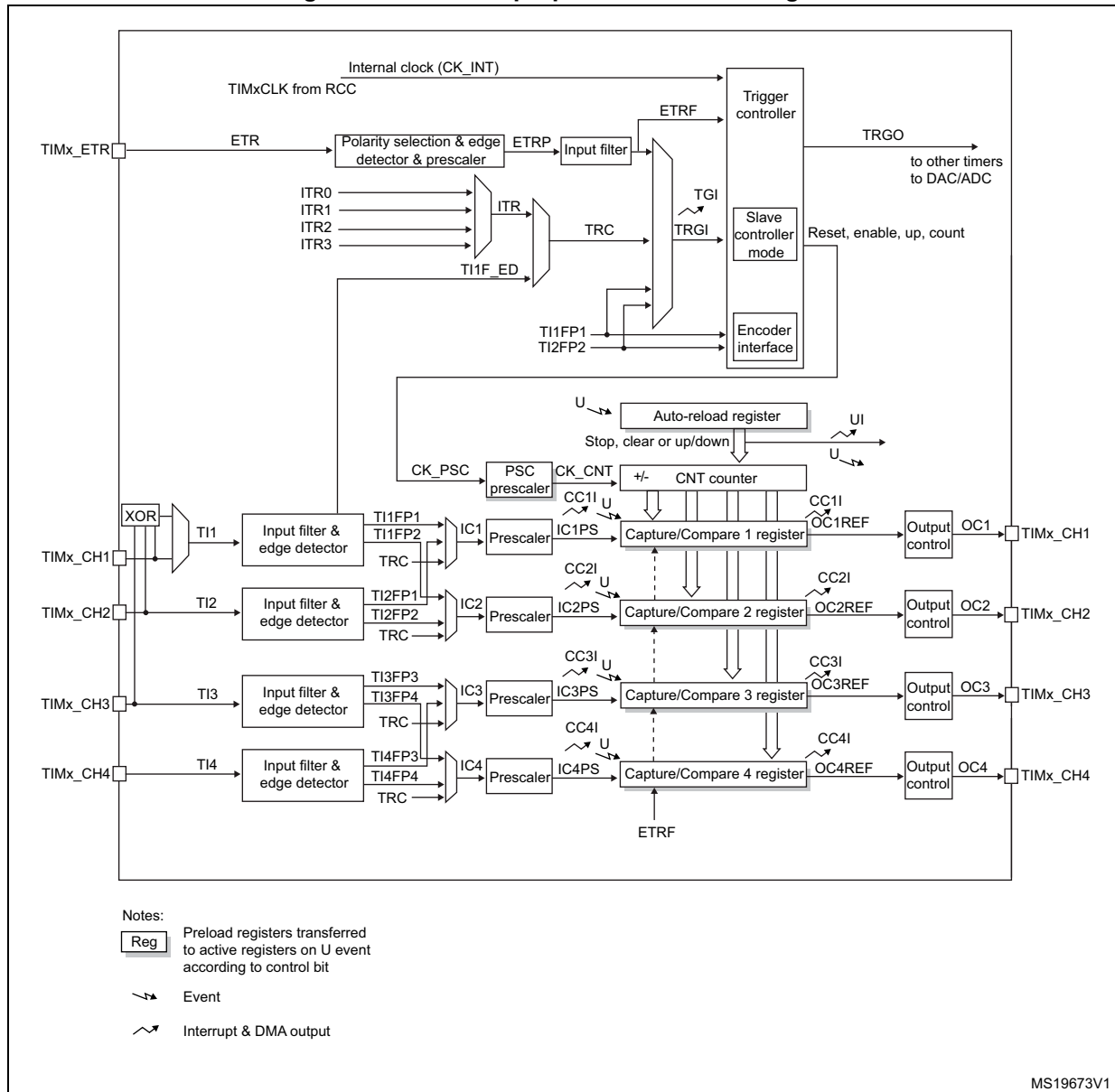
The timers are completely independent, and do not share any resources. They can be synchronized together as described in [Section 35.3.19: Timer synchronization](#).

### 35.2 TIM2/TIM3/TIM4/TIM5 main features

General-purpose TIMx timer features include:

- 16-bit (TIM3, TIM4) or 32-bit (TIM2 and TIM5) up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 316. General-purpose timer block diagram



MS19673V1

## 35.3 TIM2/TIM3/TIM4/TIM5 functional description

### 35.3.1 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related auto-reload register. The counter can count up, down or both up and down but also down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx\_CNT)
- Prescaler Register (TIMx\_PSC):
- Auto-Reload Register (TIMx\_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx\_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx\_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIMx\_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT\_EN is set 1 clock cycle after CEN.

#### Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx\_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 317* and *Figure 318* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 317. Counter timing diagram with prescaler division change from 1 to 2

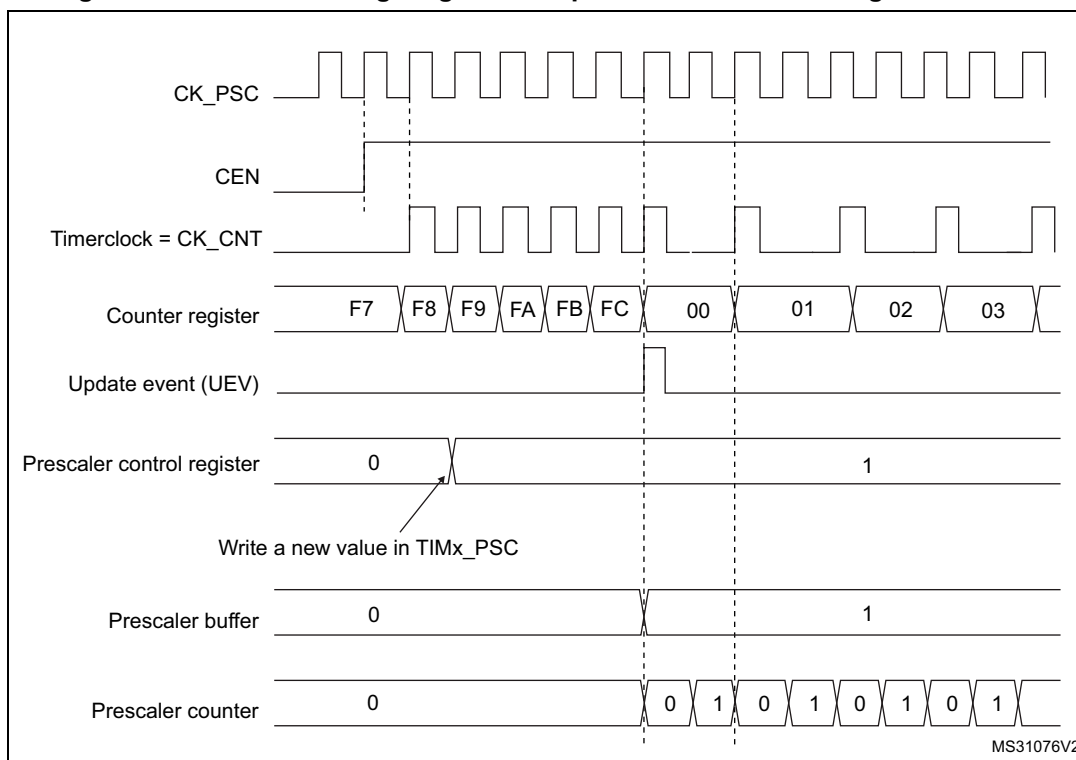
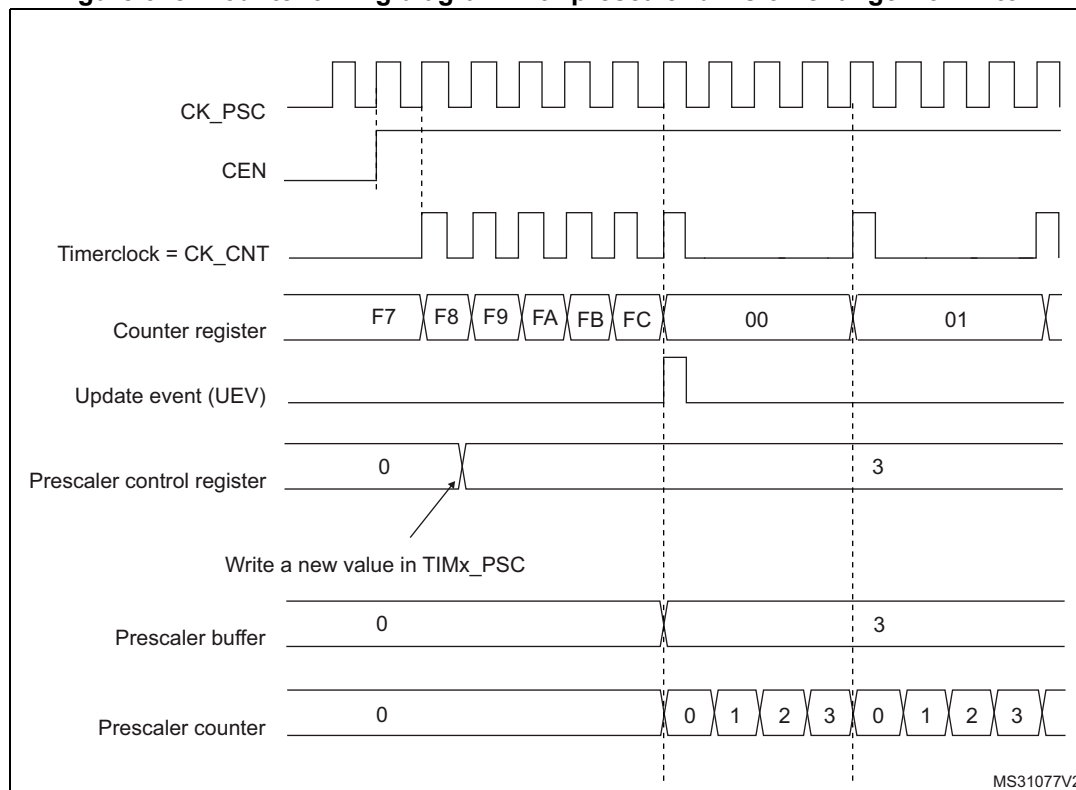


Figure 318. Counter timing diagram with prescaler division change from 1 to 4



### 35.3.2 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).

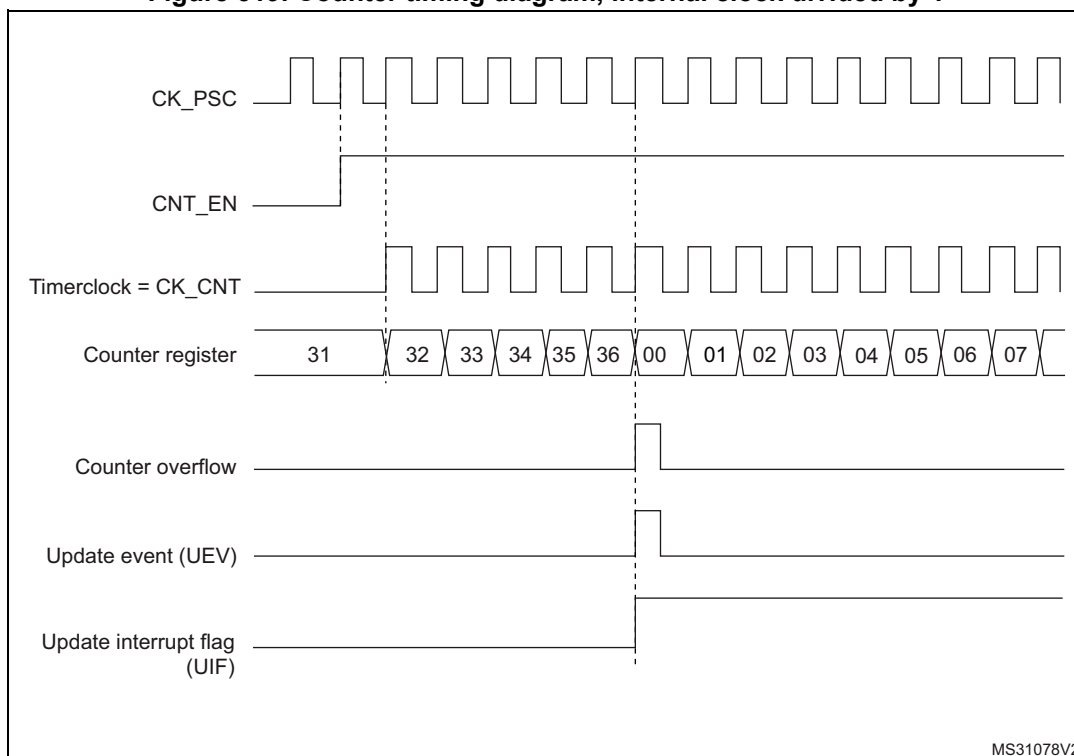
The UEV event can be disabled by software by setting the UDIS bit in TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well 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)

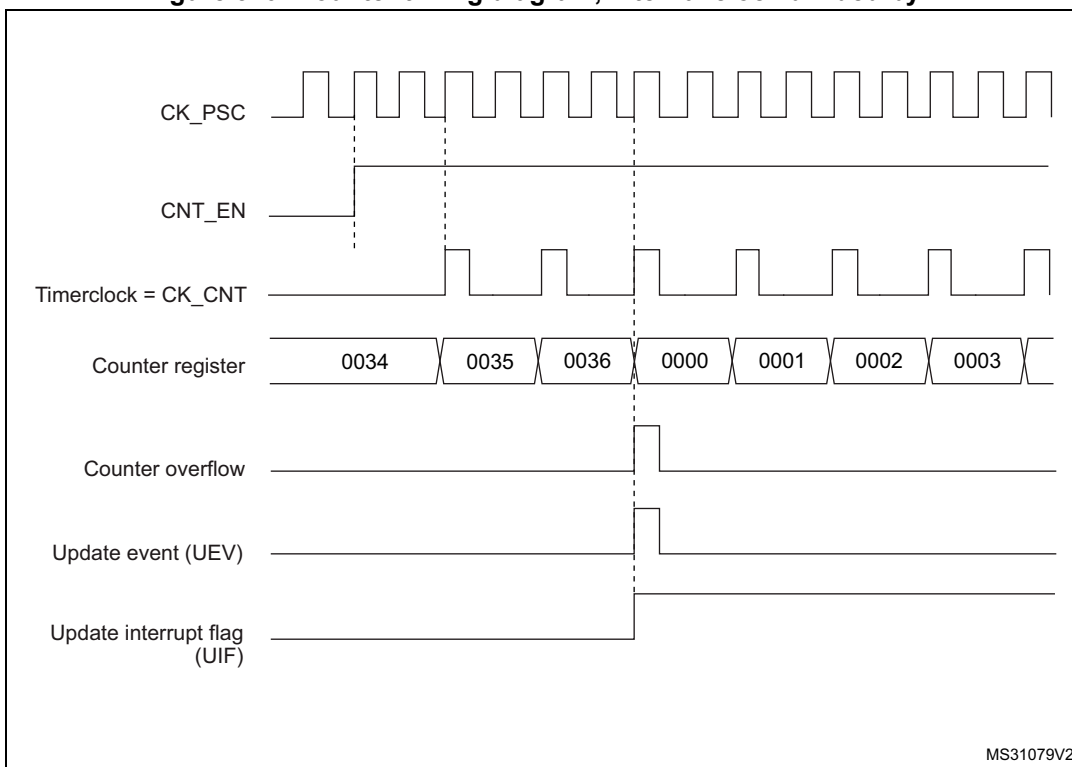
The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR=0x36.

**Figure 319. Counter timing diagram, internal clock divided by 1**



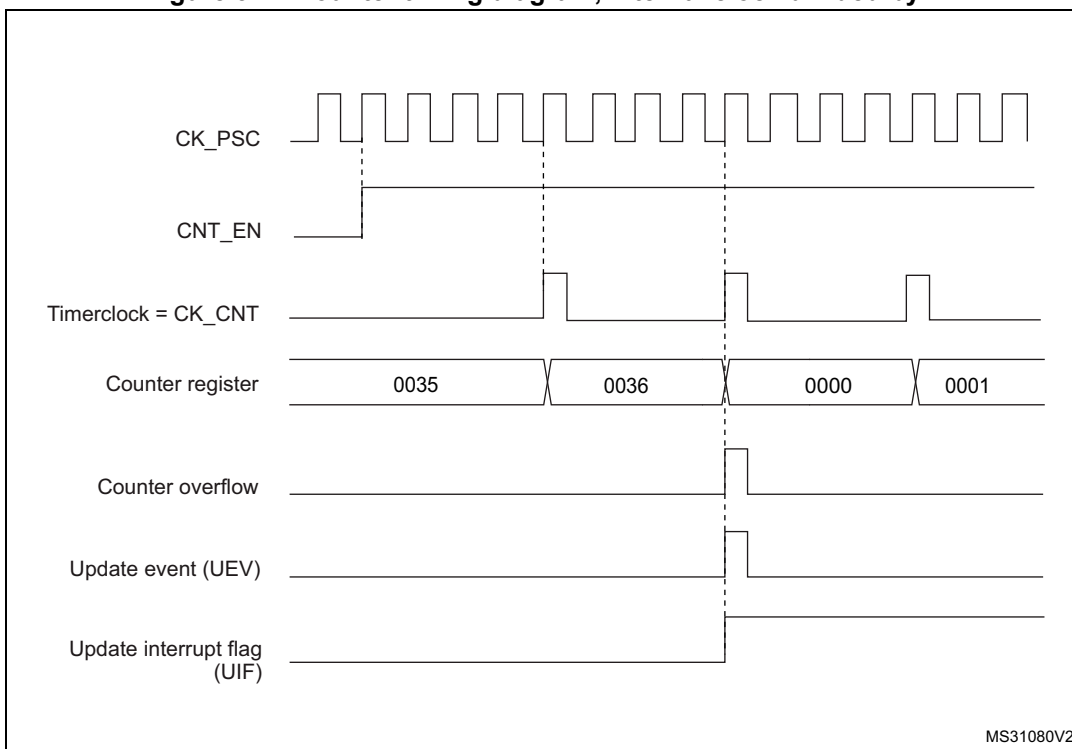
MS31078V2

Figure 320. Counter timing diagram, internal clock divided by 2



MS31079V2

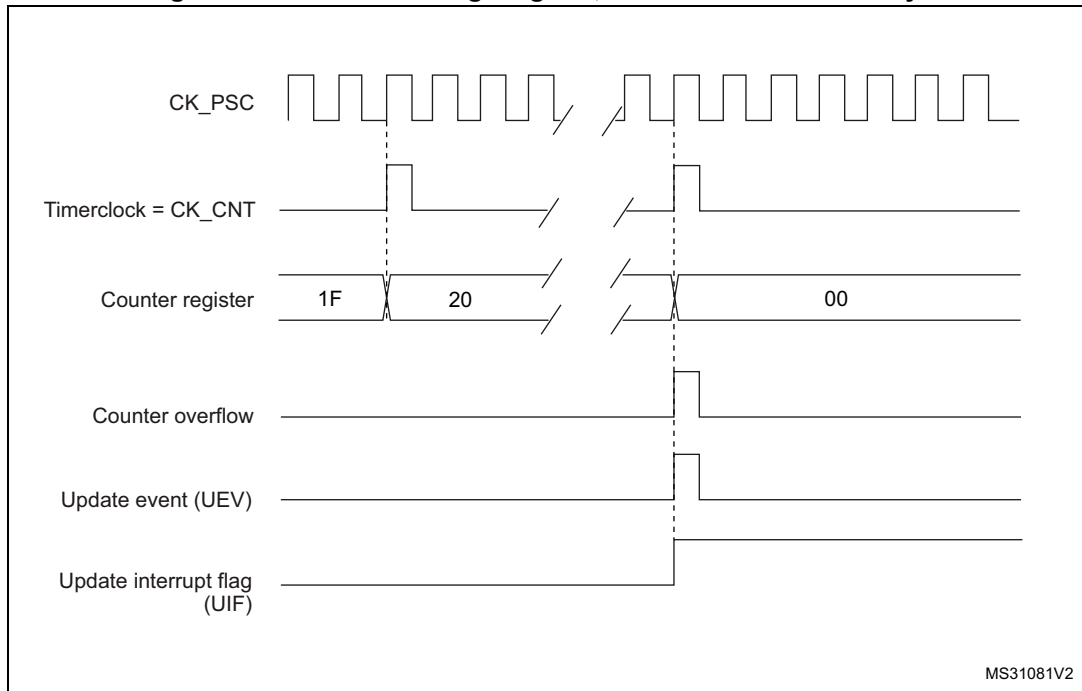
Figure 321. Counter timing diagram, internal clock divided by 4



MS31080V2

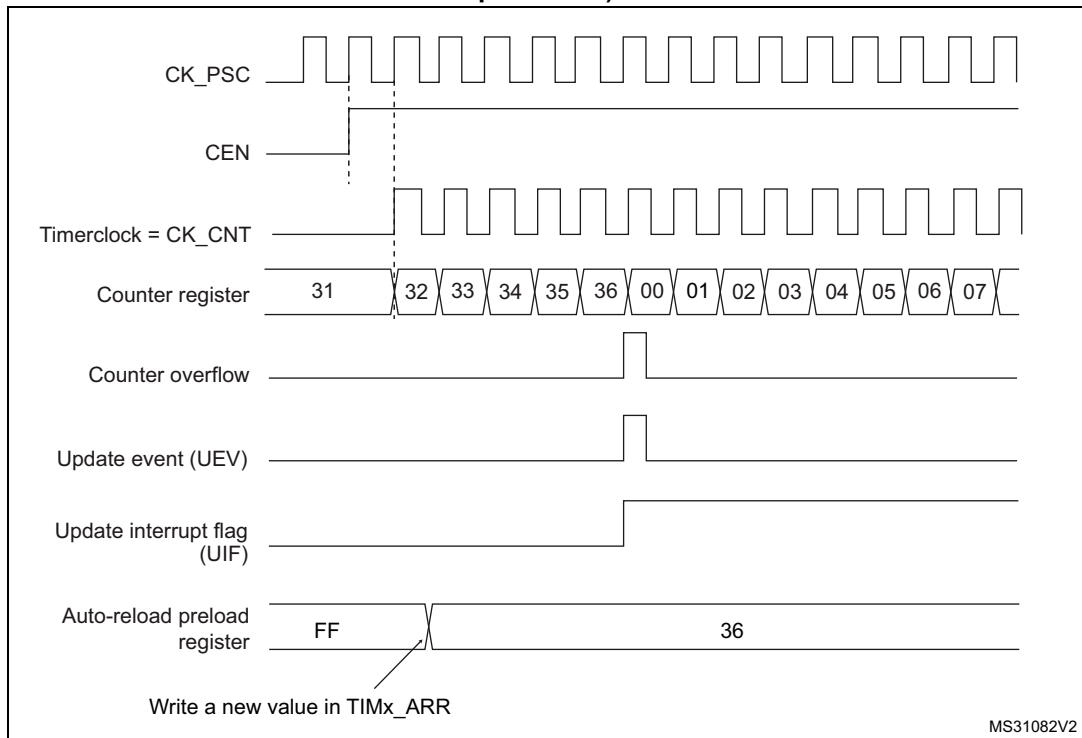


Figure 322. Counter timing diagram, internal clock divided by N



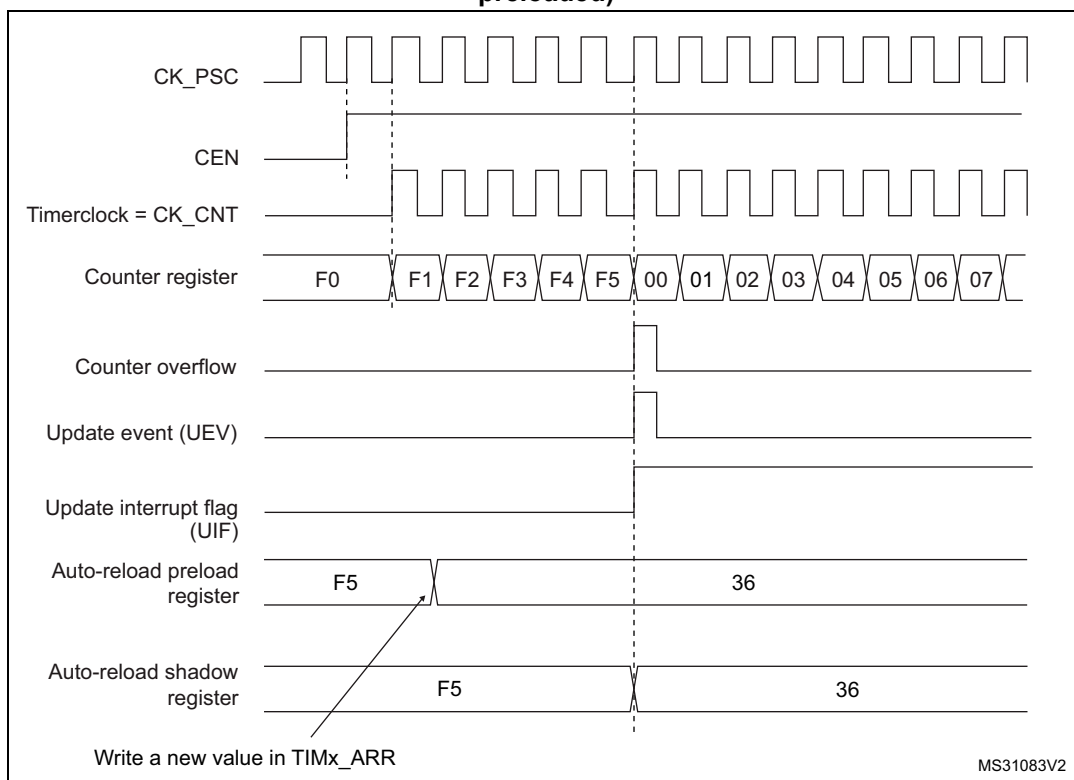
MS31081V2

Figure 323. Counter timing diagram, Update event when ARPE=0 (TIMx\_ARR not preloaded)



MS31082V2

**Figure 324. Counter timing diagram, Update event when ARPE=1 (TIMx\_ARR preloaded)**



### Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx\_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

An Update event can be generated at each counter underflow or by setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller)

The UEV update event can be disabled by software by setting the UDIS bit in TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

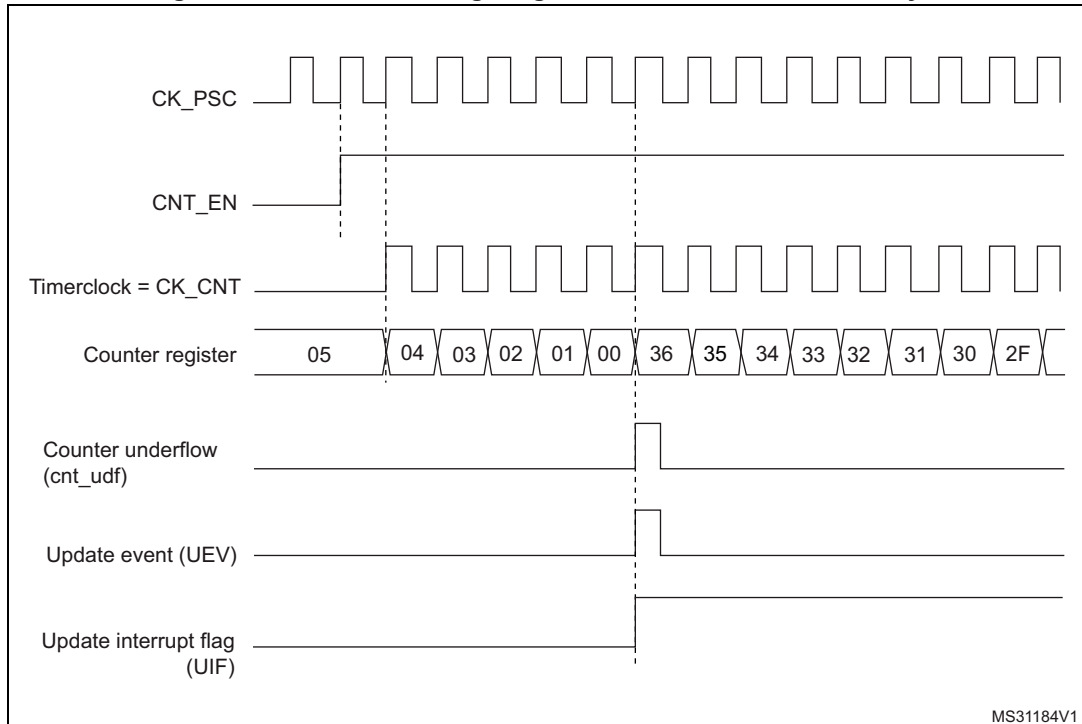
In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx\_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR=0x36.

**Figure 325. Counter timing diagram, internal clock divided by 1**



**Figure 326. Counter timing diagram, internal clock divided by 2**

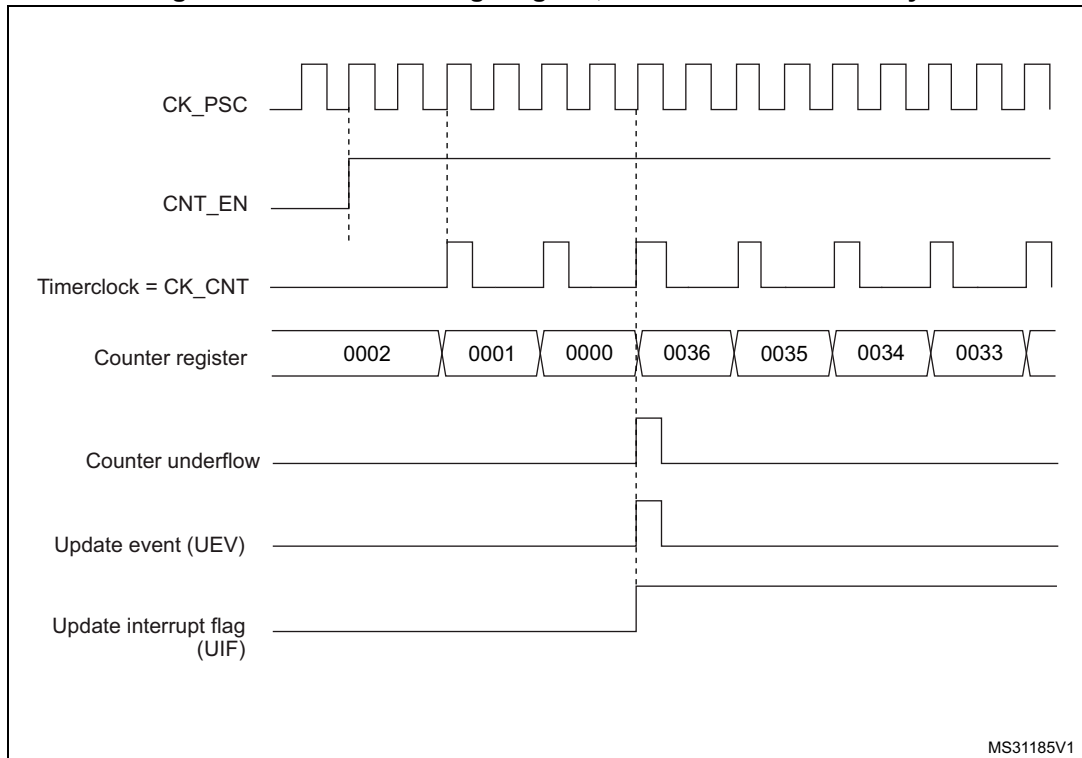


Figure 327. Counter timing diagram, internal clock divided by 4

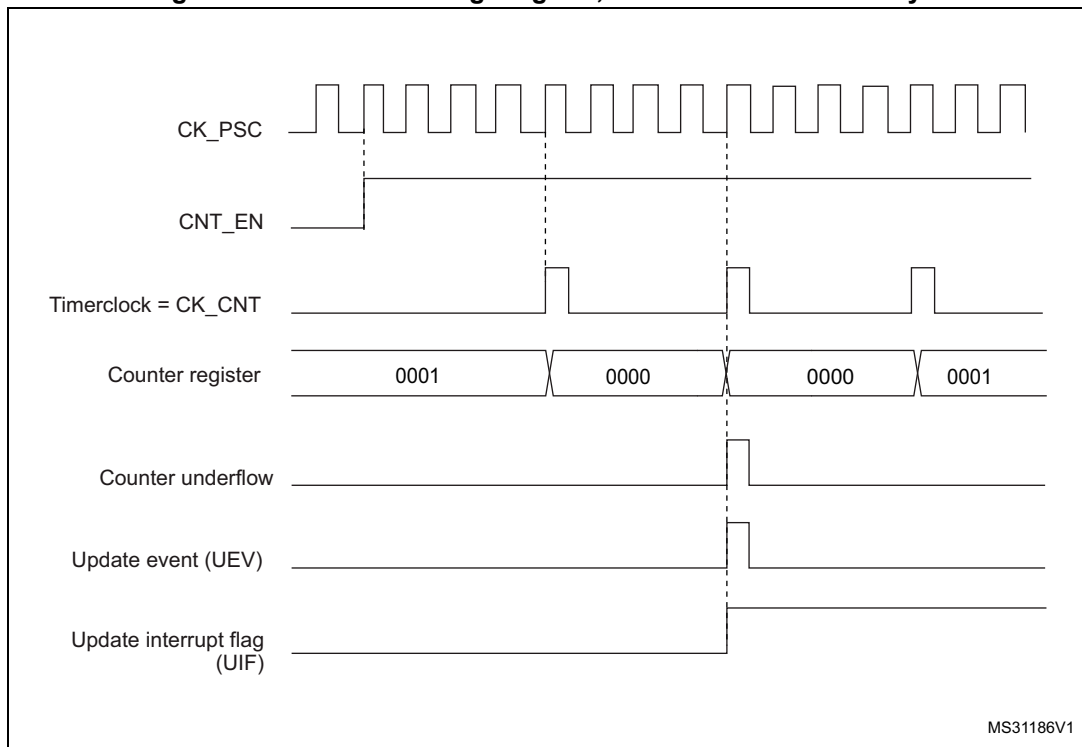
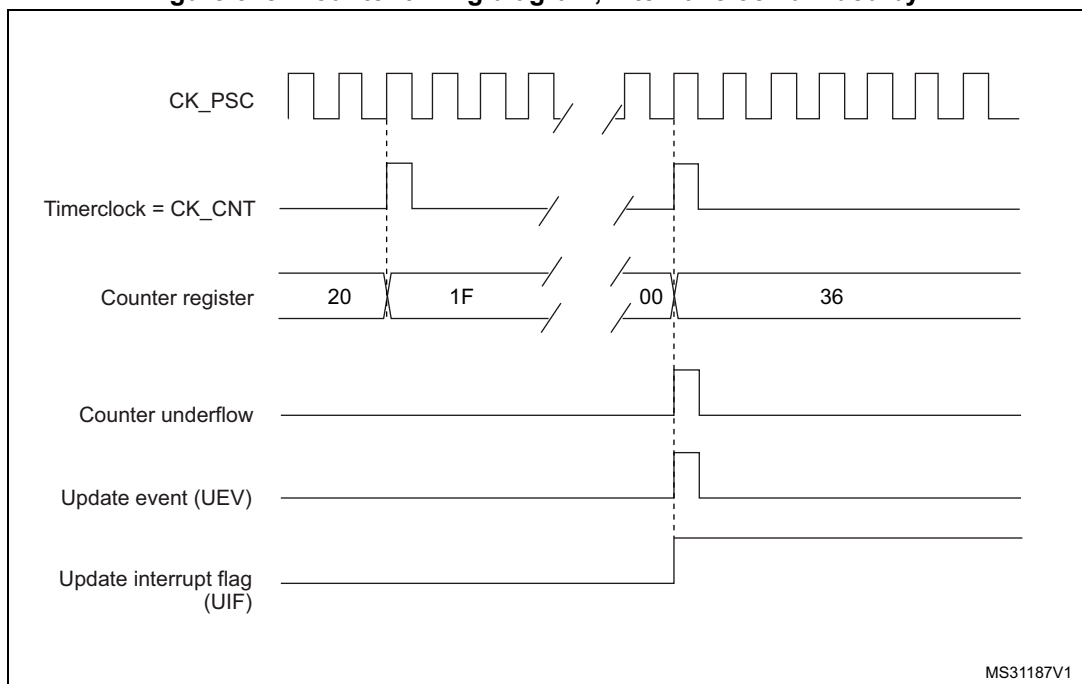
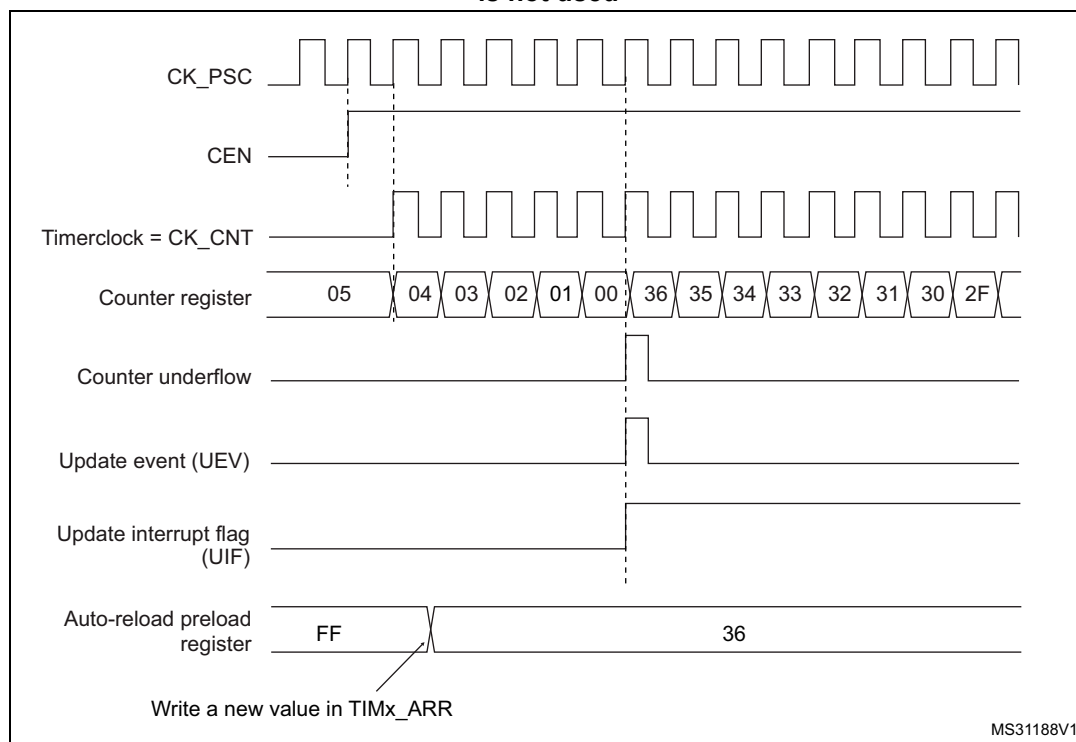


Figure 328. Counter timing diagram, internal clock divided by N



**Figure 329. 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

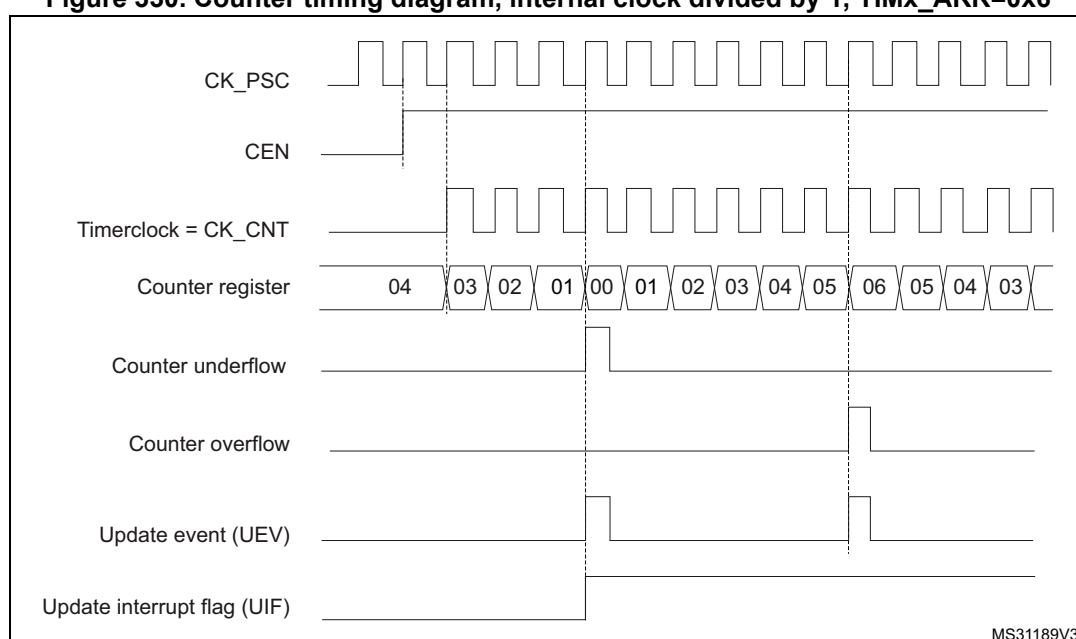
DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

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).

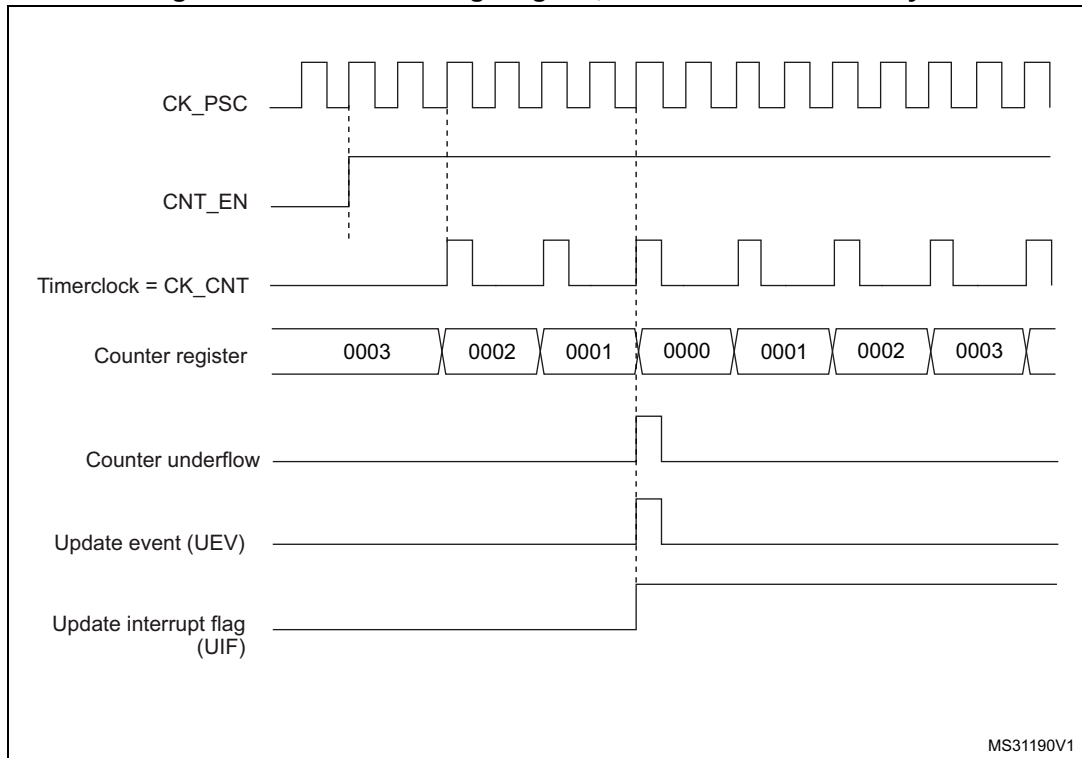
The following figures show some examples of the counter behavior for different clock frequencies.

**Figure 330. Counter timing diagram, internal clock divided by 1, TIMx\_ARR=0x6**

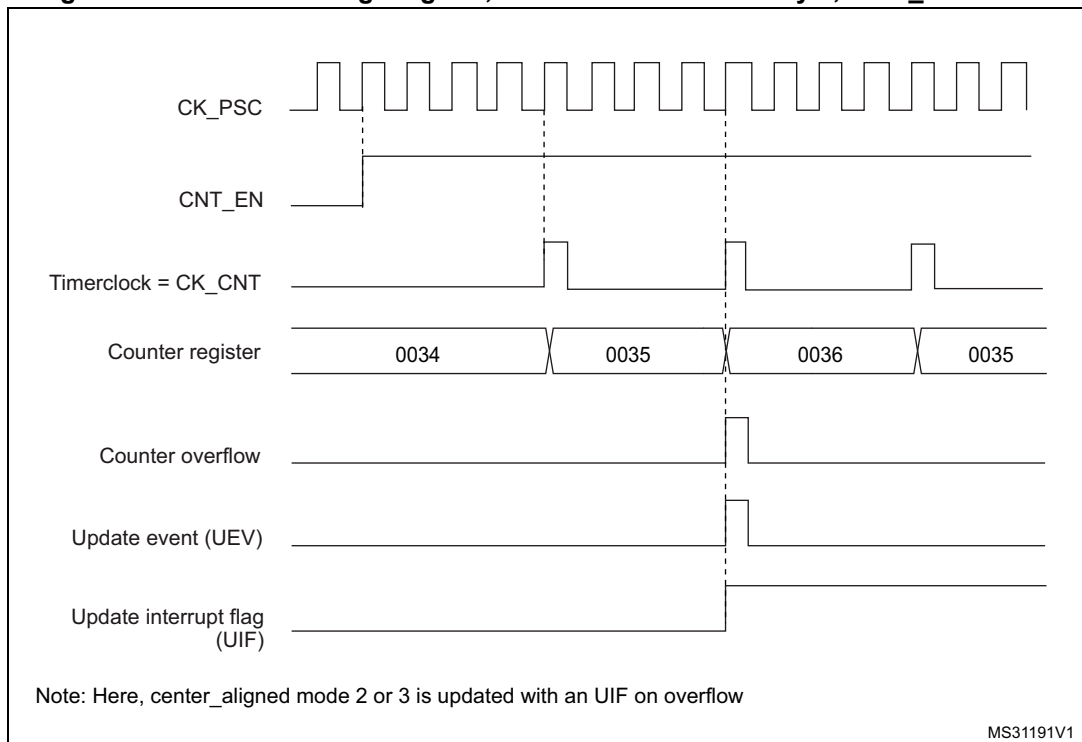


1. Here, center-aligned mode 1 is used (for more details refer to [Section 35.4.1: TIMx control register 1 \(TIMx\\_CR1\)](#) on page 1234).

**Figure 331. Counter timing diagram, internal clock divided by 2**

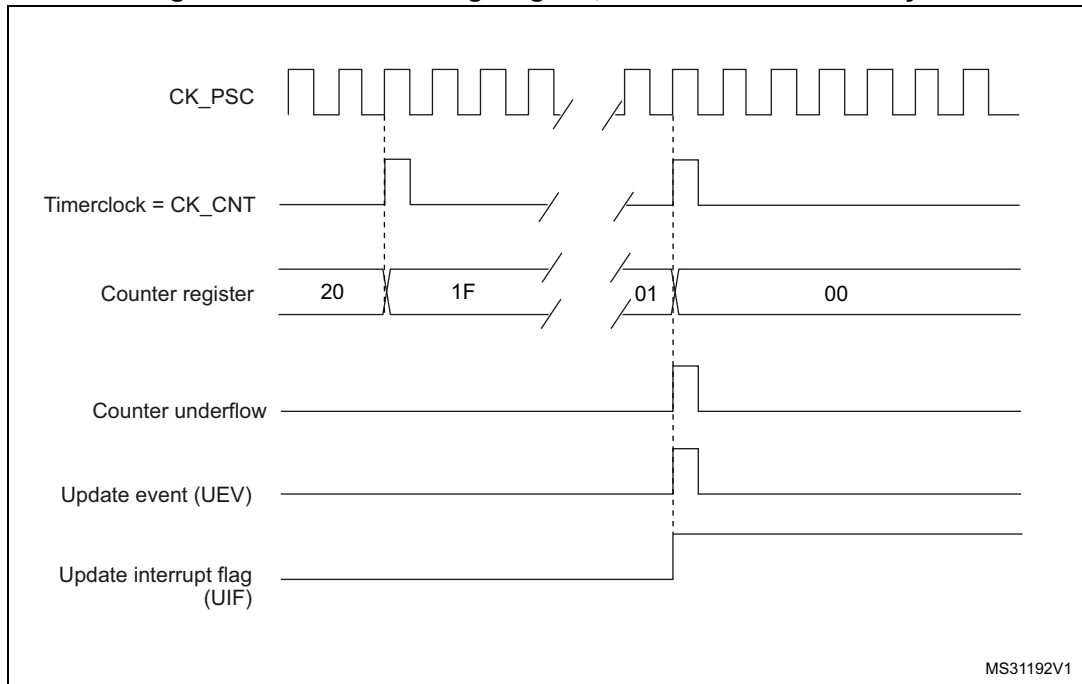


**Figure 332. Counter timing diagram, internal clock divided by 4, TIMx\_ARR=0x36**



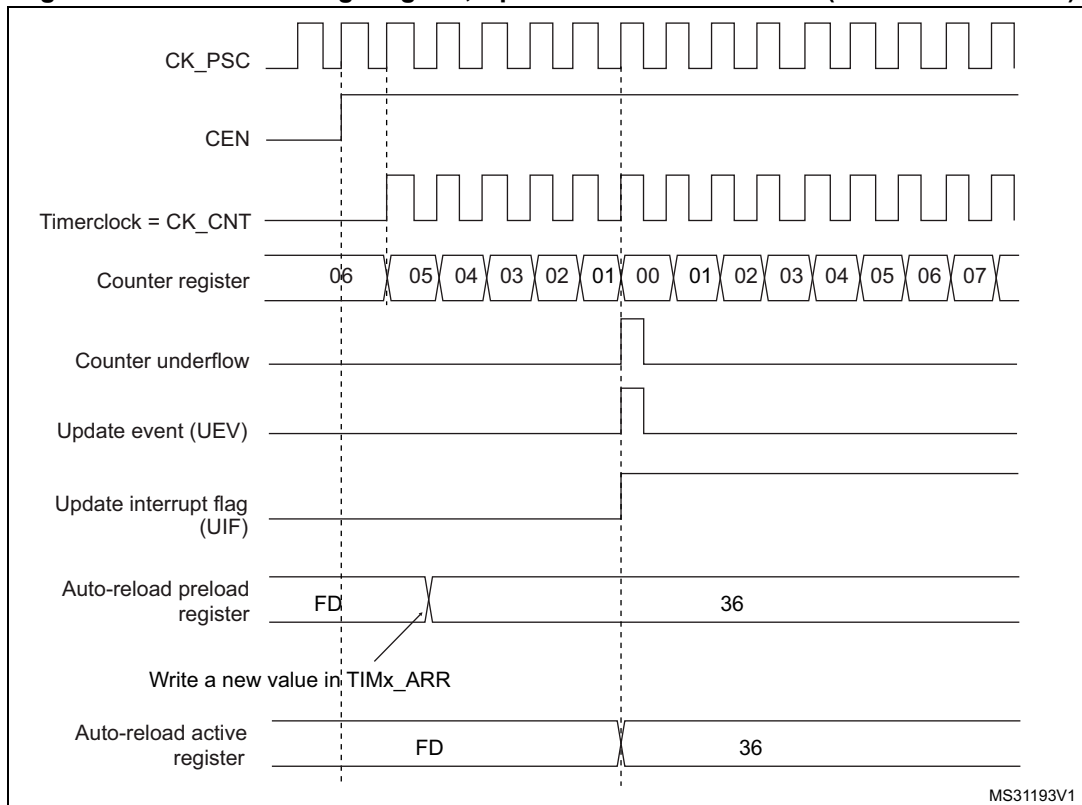
1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 333. Counter timing diagram, internal clock divided by N



MS31192V1

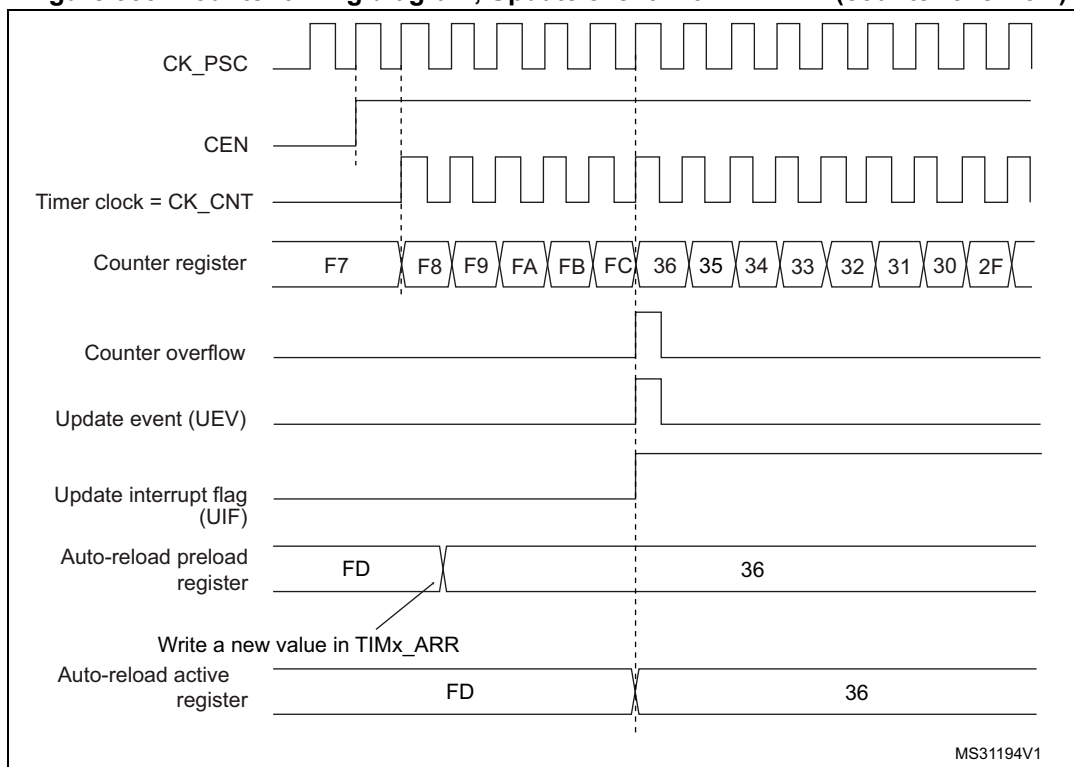
Figure 334. Counter timing diagram, Update event with ARPE=1 (counter underflow)



MS31193V1



Figure 335. Counter timing diagram, Update event with ARPE=1 (counter overflow)



### 35.3.3 Clock selection

The counter clock can be provided by the following clock sources:

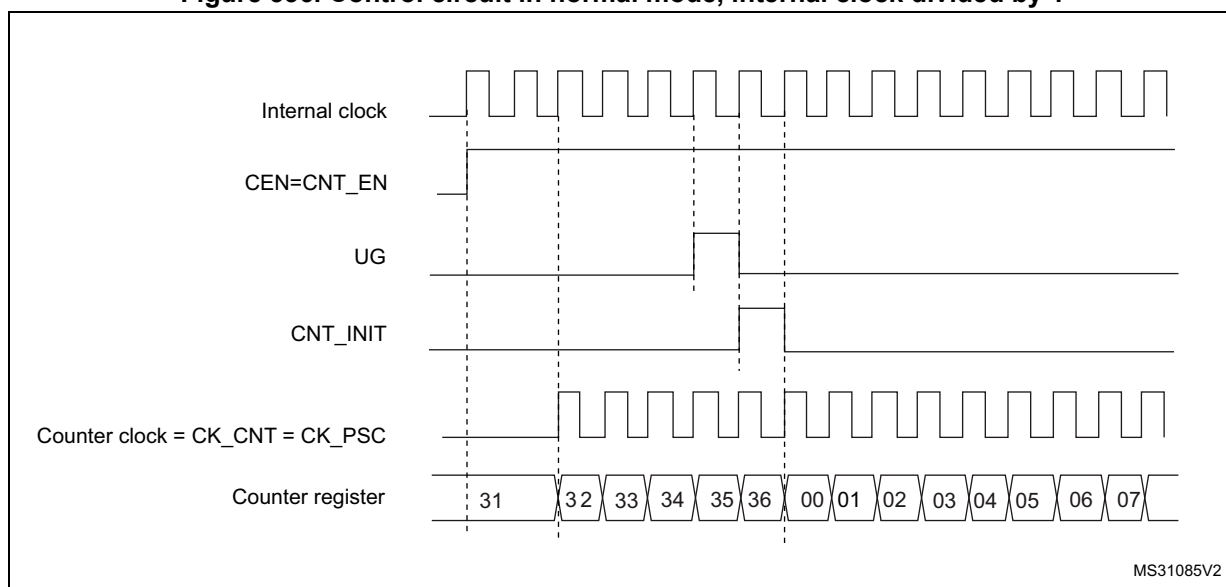
- Internal clock (CK\_INT)
- External clock mode1: external input pin (TIx)
- External clock mode2: external trigger input (ETR)
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, you can configure Timer 13 to act as a prescaler for Timer 2. Refer to : [Using one timer as prescaler for another timer on page 1229](#) 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 336](#) shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

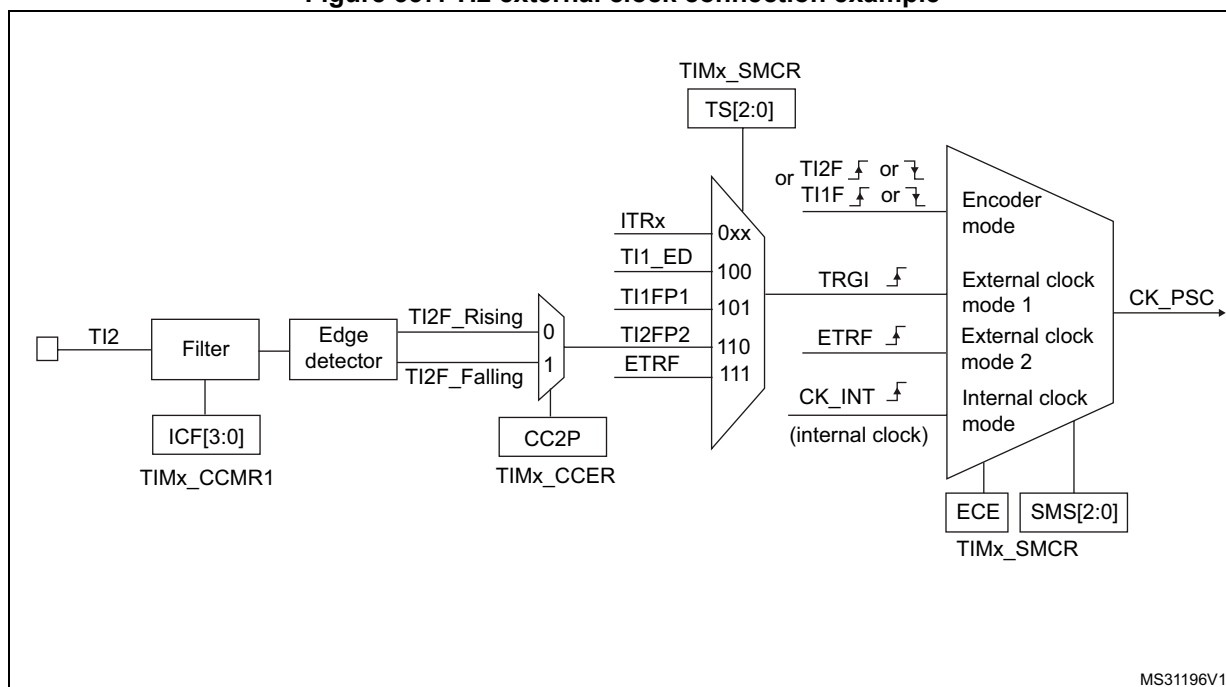
Figure 336. Control circuit in normal mode, internal clock divided by 1



**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 337. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

For example, to configure the upcounter to count in response to a falling edge on the TI2 input, use the following procedure:

1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S= '01 in the TIMx\_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx\_CCMR1 register (if no filter is needed, keep IC2F=0000).

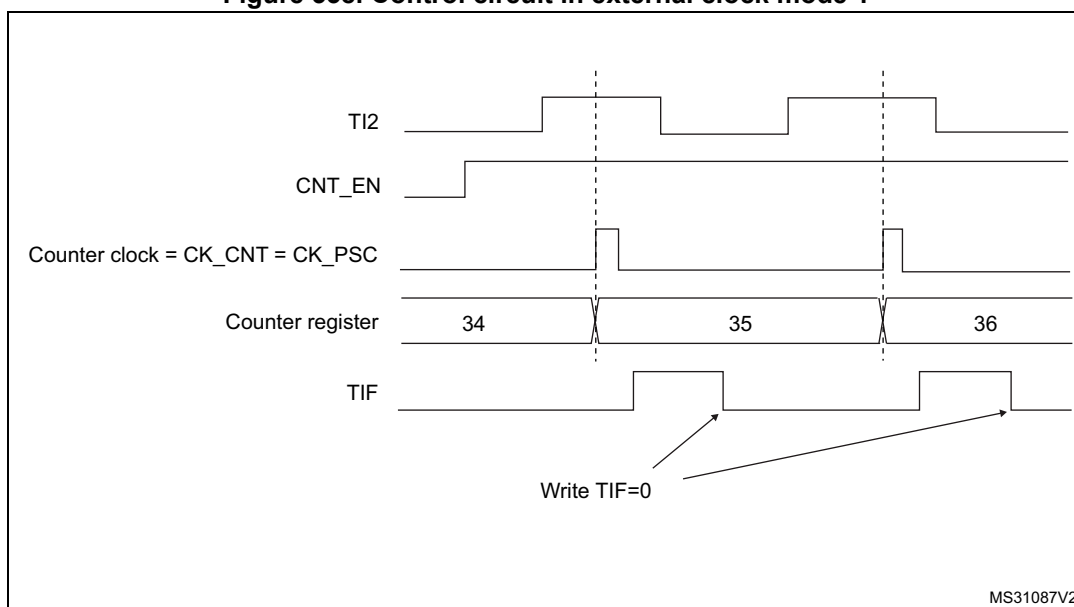
*Note:* The capture prescaler is not used for triggering, so you don't need to configure it.

3. Select rising edge polarity by writing CC2P=0 and CC2NP=0 and CC2NP=0 in the TIMx\_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx\_SMCR register.
5. Select TI2 as the input source by writing TS=110 in the TIMx\_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx\_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

**Figure 338. Control circuit in external clock mode 1**



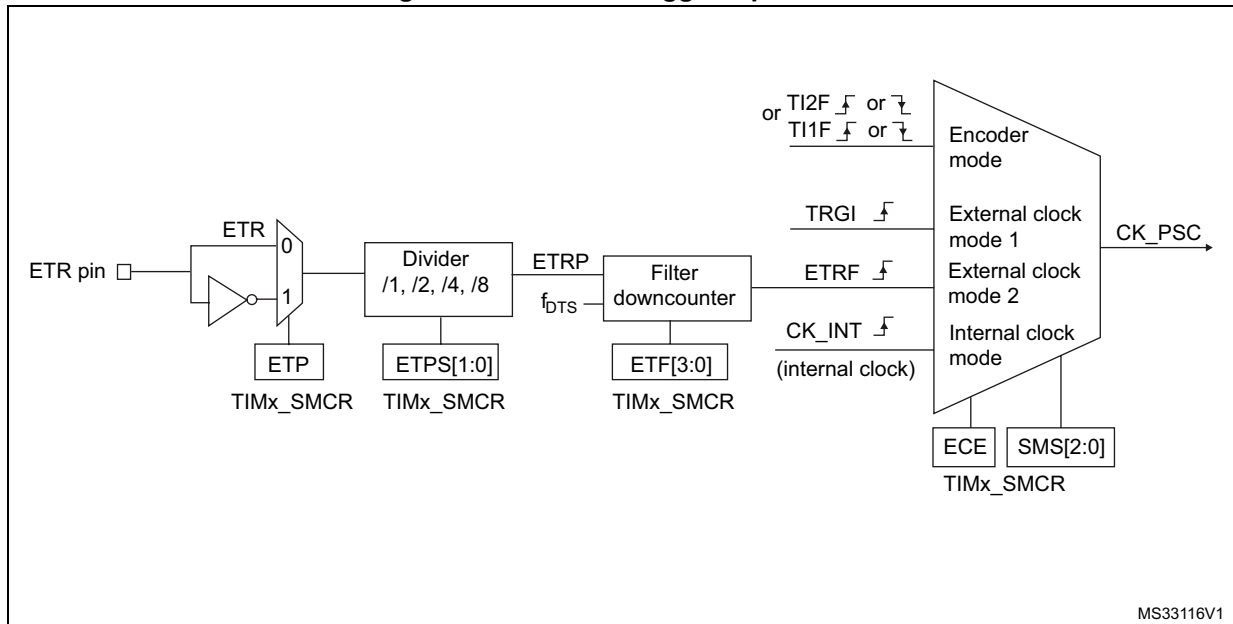
**External clock source mode 2**

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 339](#) gives an overview of the external trigger input block.

Figure 339. External trigger input block



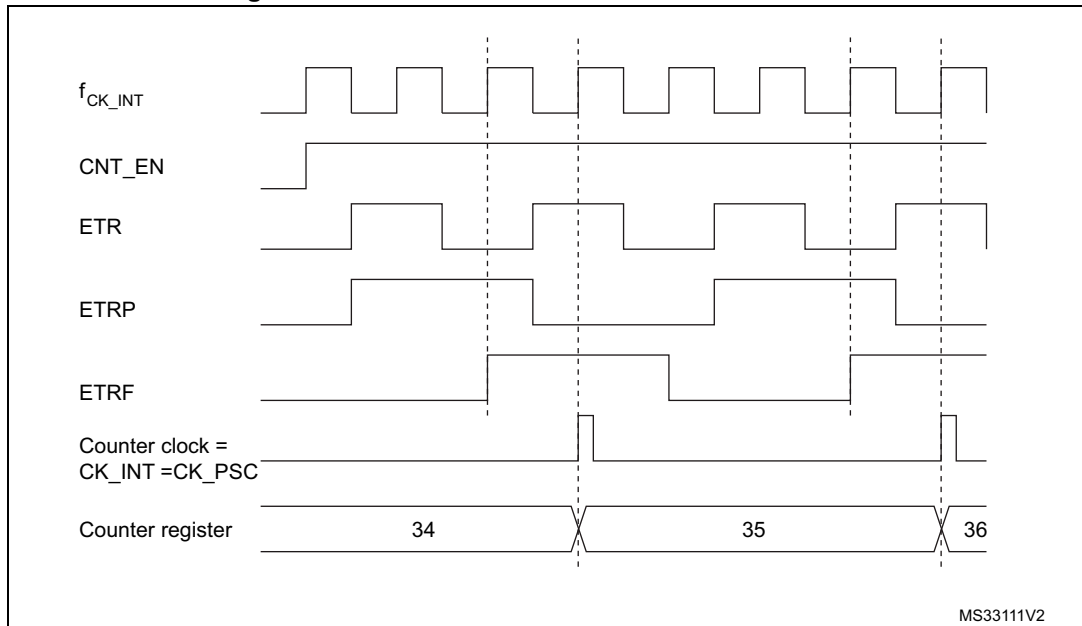
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx\_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx\_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx\_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx\_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx\_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

Figure 340. Control circuit in external clock mode 2



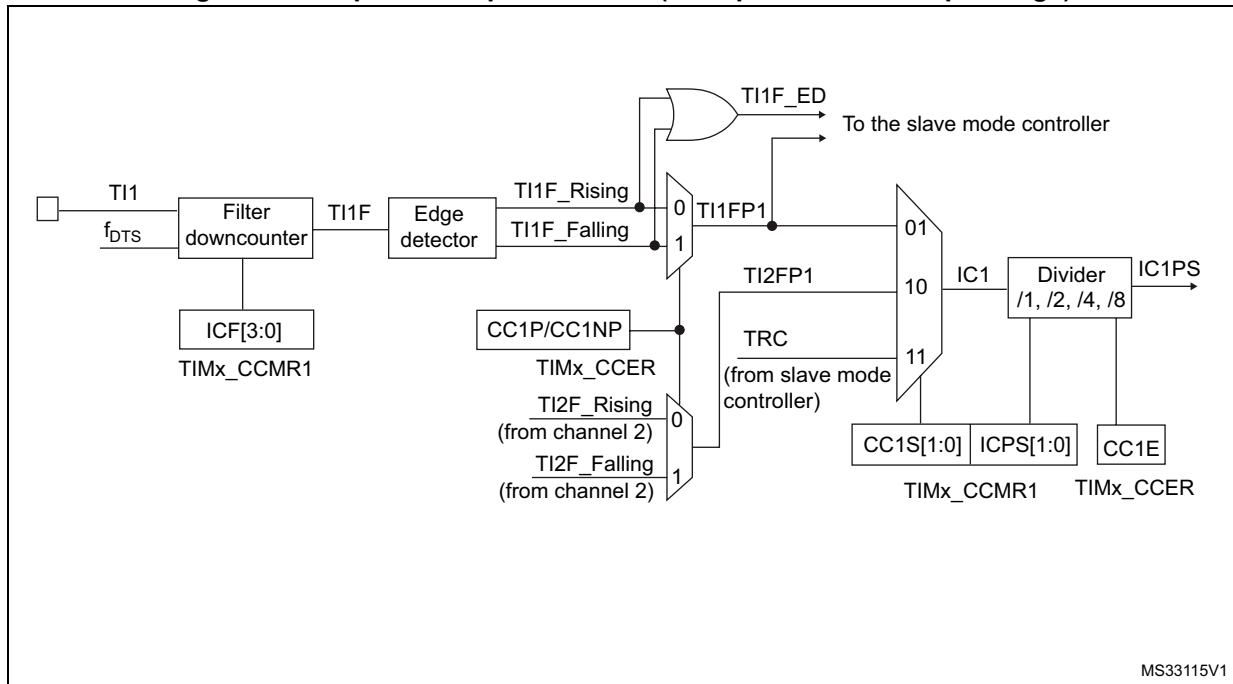
### 35.3.4 Capture/Compare channels

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 T<sub>ix</sub> input to generate a filtered signal T<sub>ixF</sub>. Then, an edge detector with polarity selection generates a signal (T<sub>ixFPx</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>xPS</sub>).

Figure 341. 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 342. Capture/Compare channel 1 main circuit

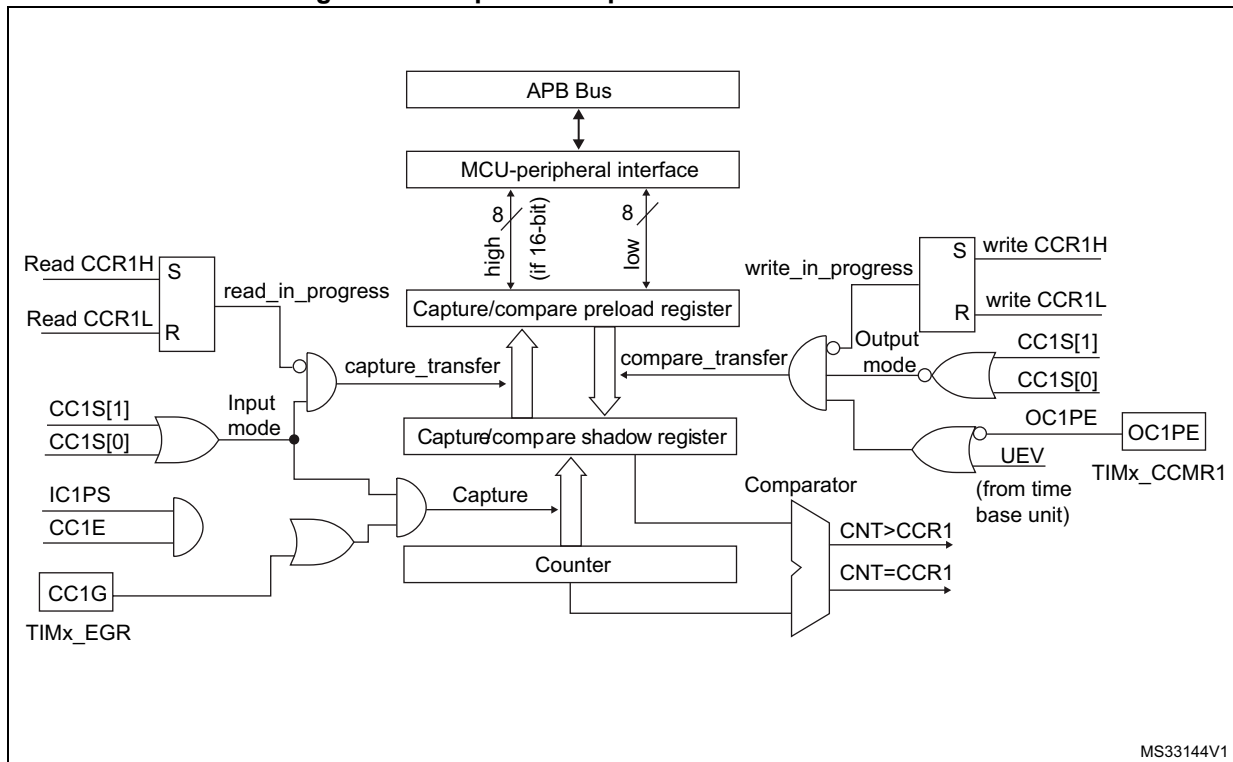
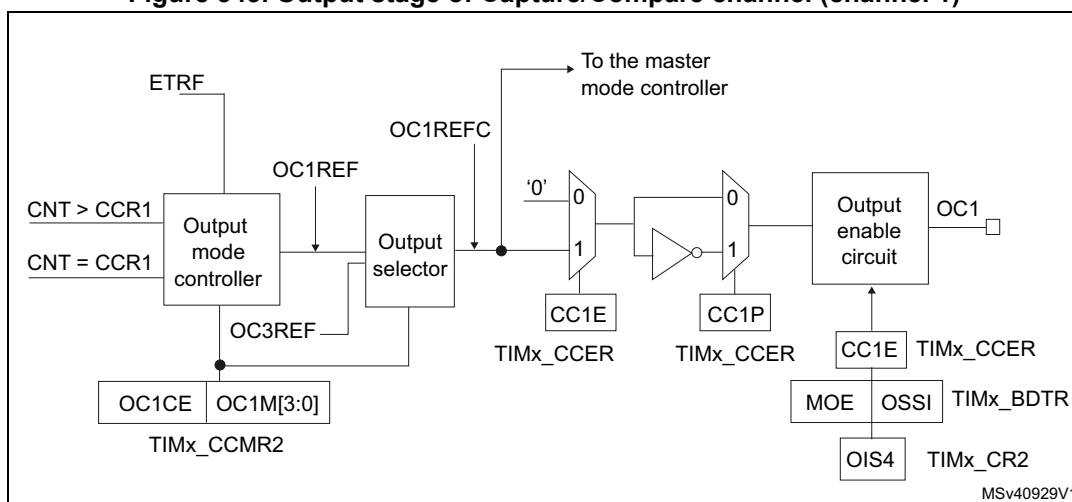


Figure 343. Output stage of Capture/Compare channel (channel 1)



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.

### 35.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers ( $TIMx\_CCRx$ ) are used to latch the value of the counter after a transition detected by the corresponding  $ICx$  signal. When a capture occurs, the corresponding  $CCXIF$  flag ( $TIMx\_SR$  register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the  $CCxIF$  flag was already high, then the over-capture flag  $CCxOF$  ( $TIMx\_SR$  register) is set.  $CCxIF$  can be cleared by software by writing it to 0 or by reading the captured data stored in the  $TIMx\_CCRx$  register.  $CCxOF$  is cleared when you write it to 0.

The following example shows how to capture the counter value in  $TIMx\_CCR1$  when  $T11$  input rises. To do this, use the following procedure:

1. Select the active input:  $TIMx\_CCR1$  must be linked to the  $T11$  input, so write the  $CC1S$  bits to 01 in the  $TIMx\_CCMR1$  register. As soon as  $CC1S$  becomes different from 00, the channel is configured in input and the  $TIMx\_CCR1$  register becomes read-only.
2. Program the input filter duration you need with respect to the signal you connect to the timer (when the input is one of the  $Tix$  ( $ICxF$  bits in the  $TIMx\_CCMRx$  register)). Let's imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on  $T11$  when 8 consecutive samples with the new level have been

detected (sampled at  $f_{DTS}$  frequency). Then write IC1F bits to 0011 in the TIMx\_CCMR1 register.

3. Select the edge of the active transition on the TI1 channel by writing the CC1P and CC1NP and CC1NP bits to 000 in the TIMx\_CCER register (rising edge in this case).
4. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx\_CCMR1 register).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx\_CCER register.
6. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx\_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx\_DIER register.

When an input capture occurs:

- The TIMx\_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

*Note:* IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx\_EGR register.

### 35.3.6 PWM input mode

This mode is a particular case of input capture mode. The procedure is the same except:

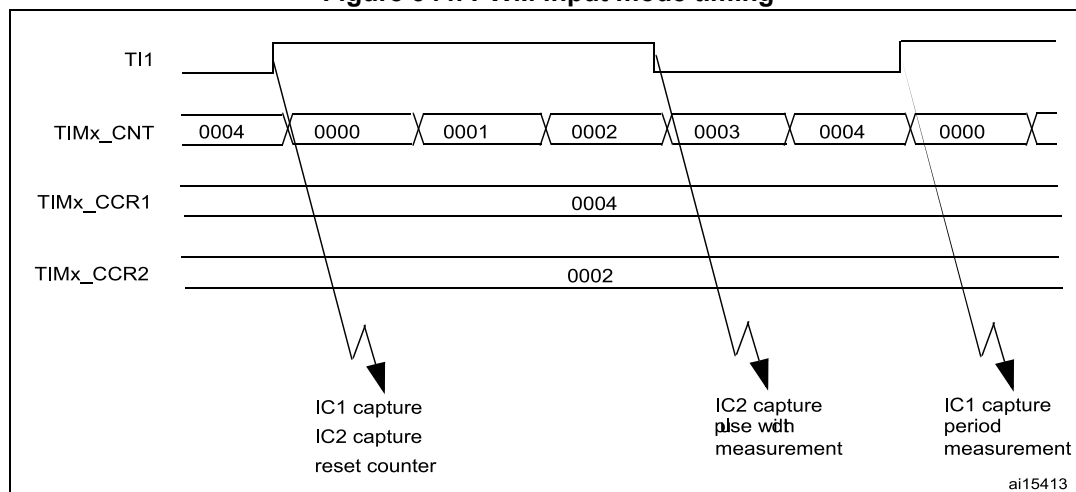
- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.



For example, you can measure the period (in TIMx\_CCR1 register) and the duty cycle (in TIMx\_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK\_INT frequency and prescaler value):

1. Select the active input for TIMx\_CCR1: write the CC1S bits to 01 in the TIMx\_CCMR1 register (TI1 selected).
2. Select the active polarity for TI1FP1 (used both for capture in TIMx\_CCR1 and counter clear): write the CC1P to '0' and the CC1NP bit to '0' (active on rising edge).
3. Select the active input for TIMx\_CCR2: write the CC2S bits to 10 in the TIMx\_CCMR1 register (TI1 selected).
4. Select the active polarity for TI1FP2 (used for capture in TIMx\_CCR2): write the CC2P bit to '1' and the CC2NP bit to '0' (active on falling edge).
5. Select the valid trigger input: write the TS bits to 101 in the TIMx\_SMCR register (TI1FP1 selected).
6. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx\_SMCR register.
7. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx\_CCER register.

**Figure 344. 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.

### 35.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx\_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (ocxref/OCx) to its active level, you just need to write 101 in the OCxM bits in the corresponding TIMx\_CCMRx register. Thus ocxref is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

e.g.: CCxP=0 (OCx active high) => OCx is forced to high level.

ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx\_CCMRx register.

Anyway, the comparison between the TIMx\_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare Mode section.

### 35.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx\_CCMRx register) and the output polarity (CCxP bit in the TIMx\_CCER register). The output pin can keep its level (OCxM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx\_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCxIE bit in the TIMx\_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx\_DIER register, CCDS bit in the TIMx\_CR2 register for the DMA request selection).

The TIMx\_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx\_CCMRx register.

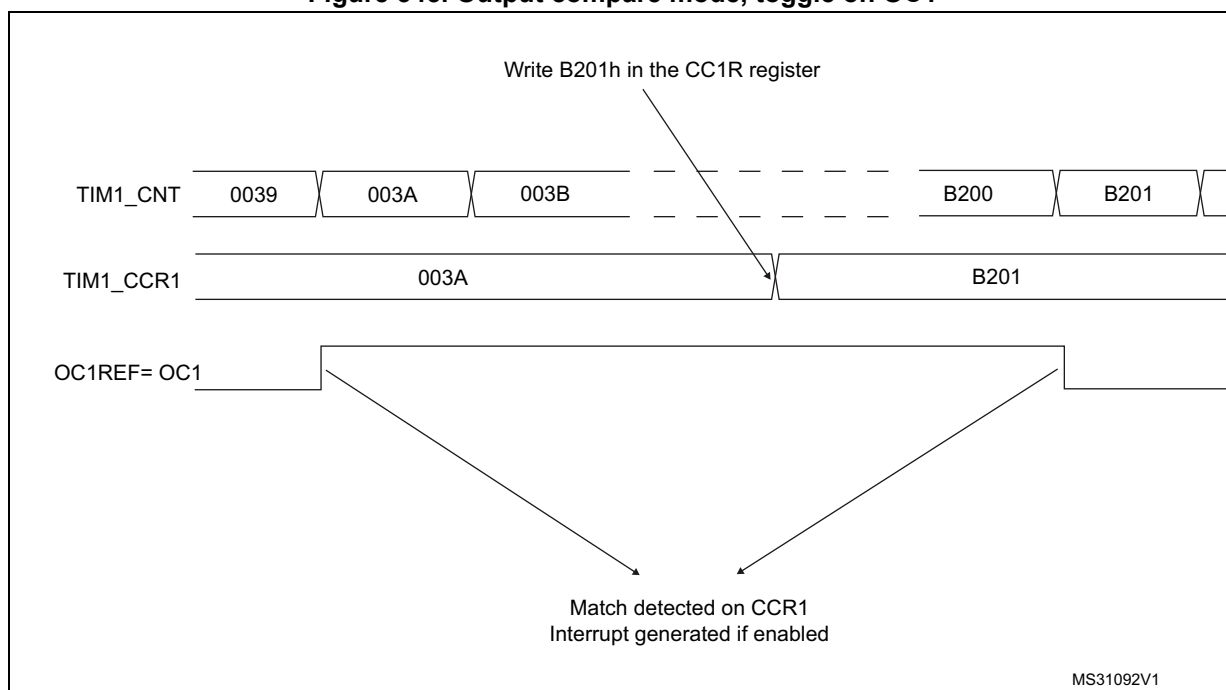
In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

#### Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx\_ARR and TIMx\_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example, you must write OCxM=011, OCxPE=0, CCxP=0 and CCxE=1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx\_CR1 register.

The TIMx\_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=0, else TIMx\_CCRx shadow register is updated only at the next update event UEV). An example is given in [Figure 345](#).

Figure 345. Output compare mode, toggle on OC1



### 35.3.9 PWM mode

Pulse width modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx\_ARR register and a duty cycle determined by the value of the TIMx\_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing 110 (PWM mode 1) or '111 (PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx\_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx\_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx\_CCER register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the TIMx\_CCER register. Refer to the TIMx\_CCERx register description for more details.

In PWM mode (1 or 2), TIMx\_CNT and TIMx\_CCRx are always compared to determine whether  $TIMx\_CCRx \leq TIMx\_CNT$  or  $TIMx\_CNT \leq TIMx\_CCRx$  (depending on the direction of the counter). However, to comply with the OCREF\_CLR functionality (OCREF can be cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison or
- When the output compare mode (OCxM bits in TIMx\_CCMRx register) switches from the "frozen" configuration (no comparison, OCxM='000) to one of the PWM modes (OCxM='110 or '111).

This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx\_CR1 register.

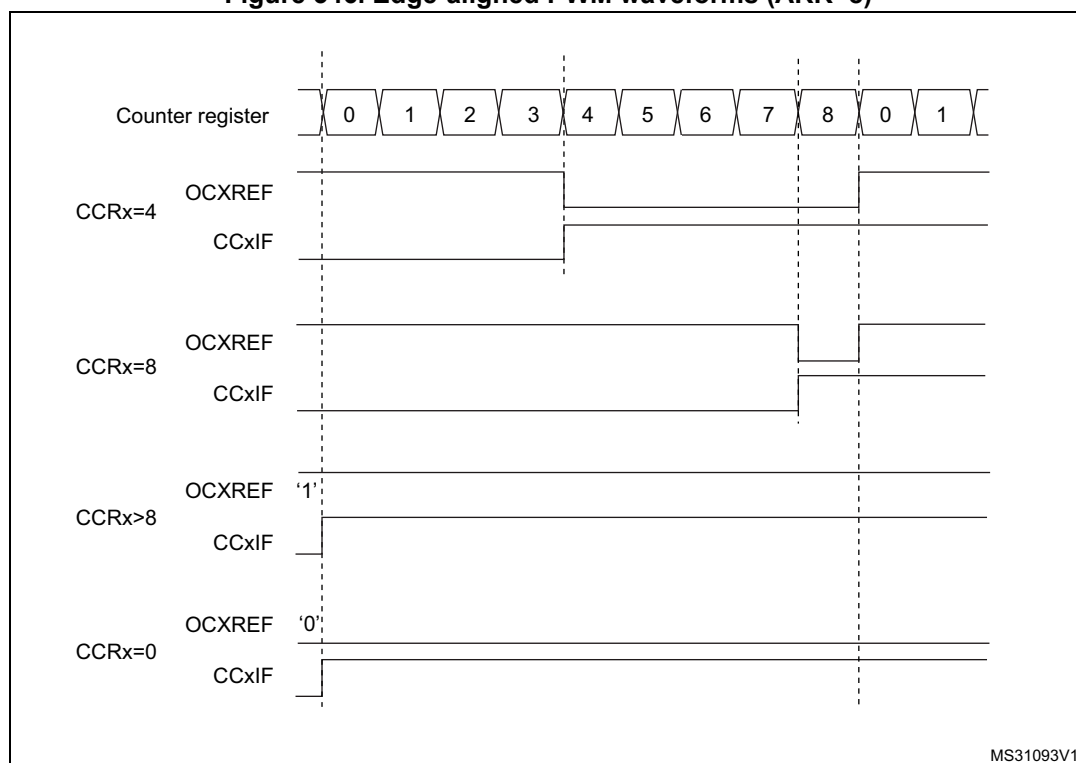
### PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx\_CR1 register is low. Refer to [Upcounting mode on page 1193](#).

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 346](#) shows some edge-aligned PWM waveforms in an example where TIMx\_ARR=8.

**Figure 346. Edge-aligned PWM waveforms (ARR=8)**



### Downcounting configuration

Downcounting is active when DIR bit in TIMx\_CR1 register is high. Refer to [Downcounting mode on page 1196](#).

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

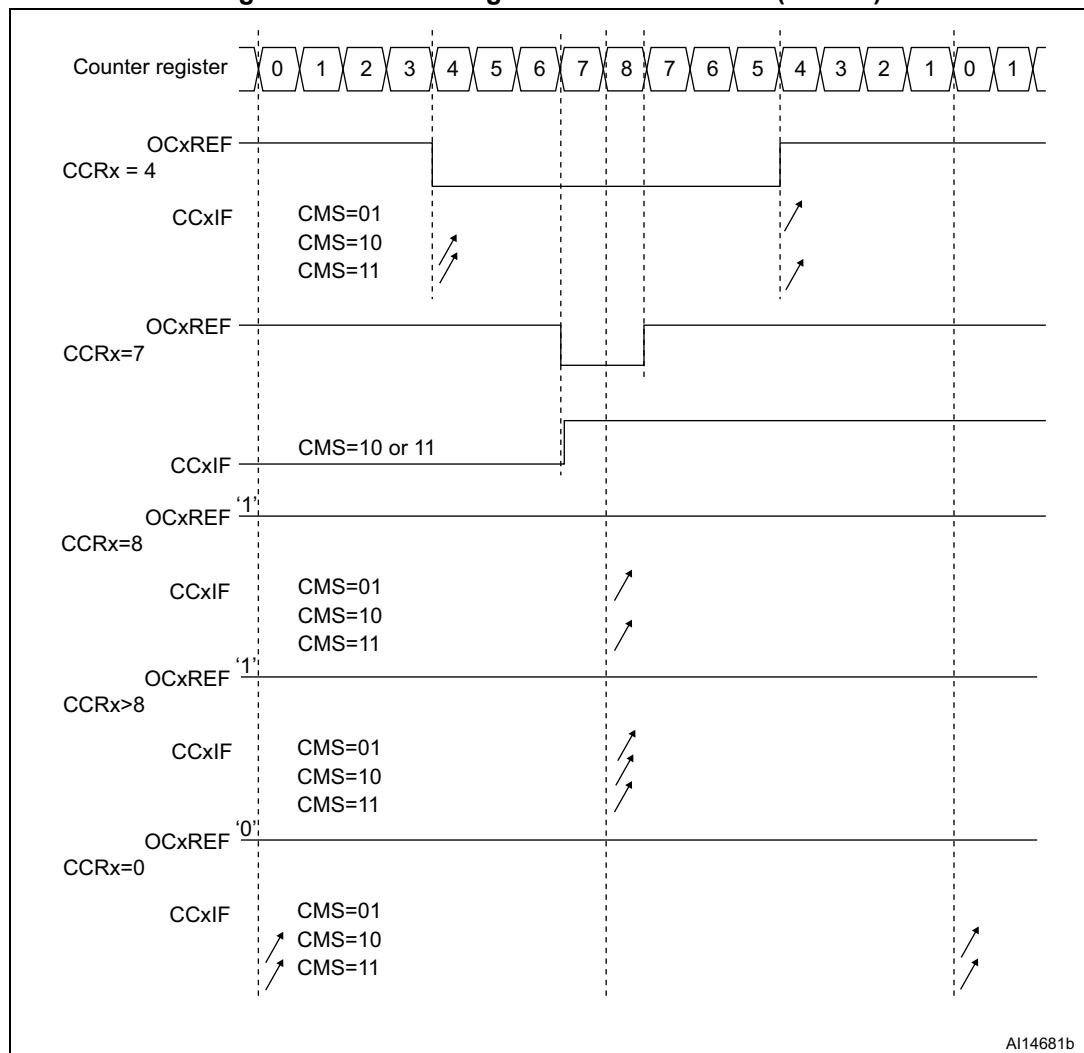
Center-aligned mode is active when the CMS bits in TIMx\_CR1 register are different from '00 (all the remaining configurations having the same effect on the ocxref/OCx signals). The

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 1199](#).

Figure 347 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 347. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIMx\_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if you write a value in the counter that is greater than the auto-reload value (TIMx\_CNT>TIMx\_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if you write 0 or write the TIMx\_ARR value in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx\_EGR register) just before starting the counter and not to write the counter while it is running.

### 35.3.10 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

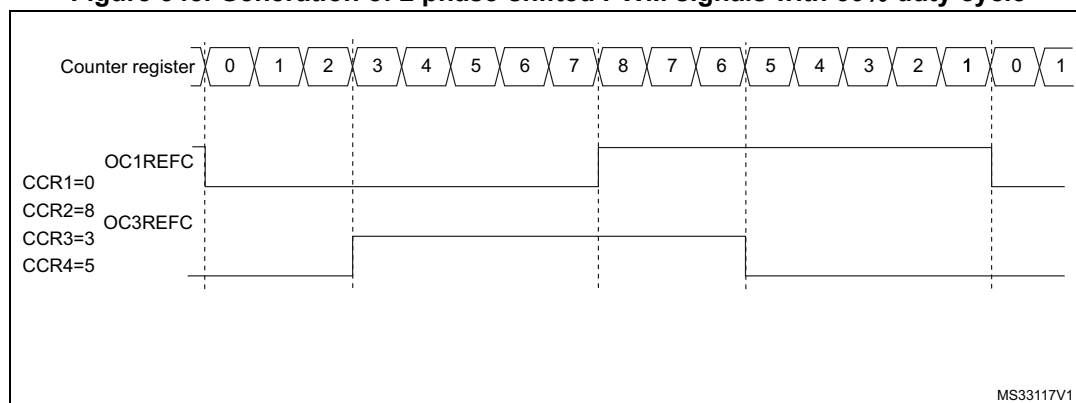
Asymmetric PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing '1110' (Asymmetric PWM mode 1) or '1111' (Asymmetric PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register.

*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 348 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 348. Generation of 2 phase-shifted PWM signals with 50% duty cycle**



### 35.3.11 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.

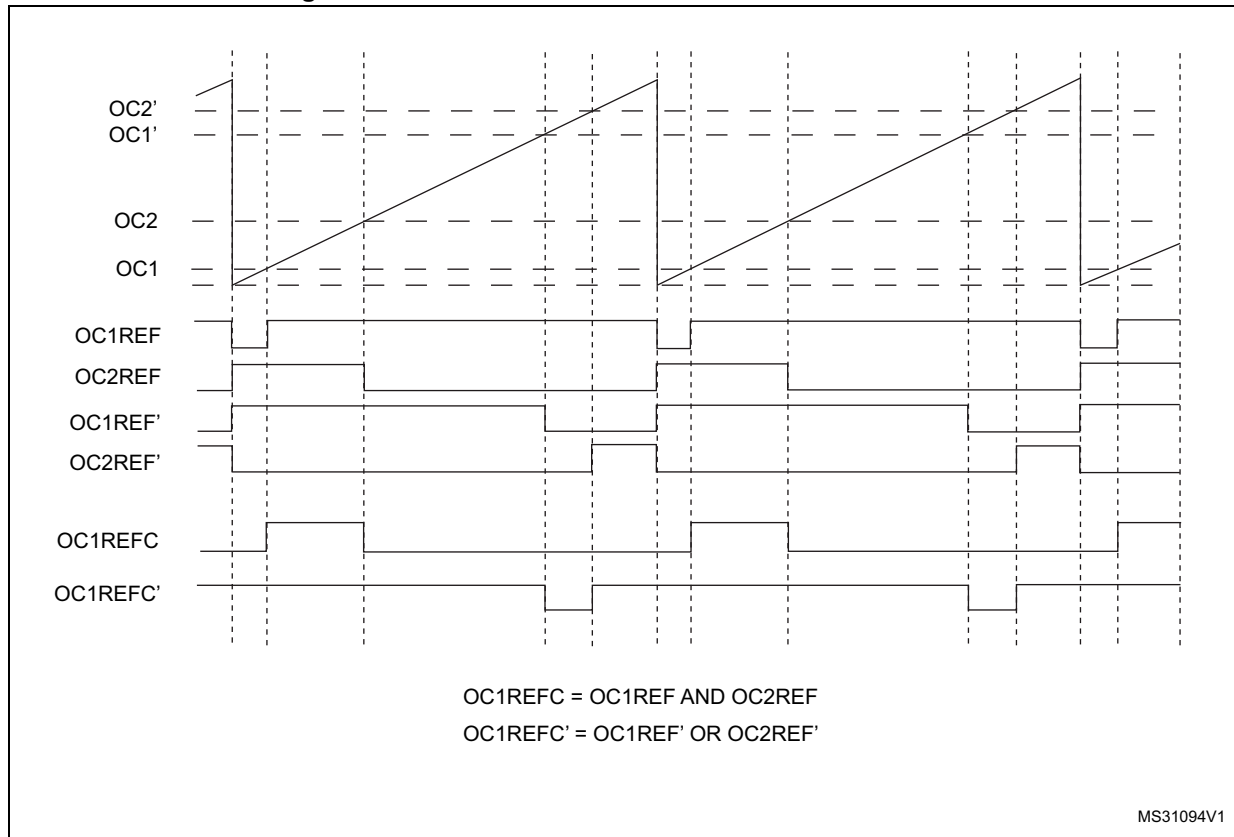
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

*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 349* 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 349. Combined PWM mode on channels 1 and 3



### 35.3.12 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.

The ocref\_clr\_int is connected to the ETRF signal (ETR after filtering).

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.

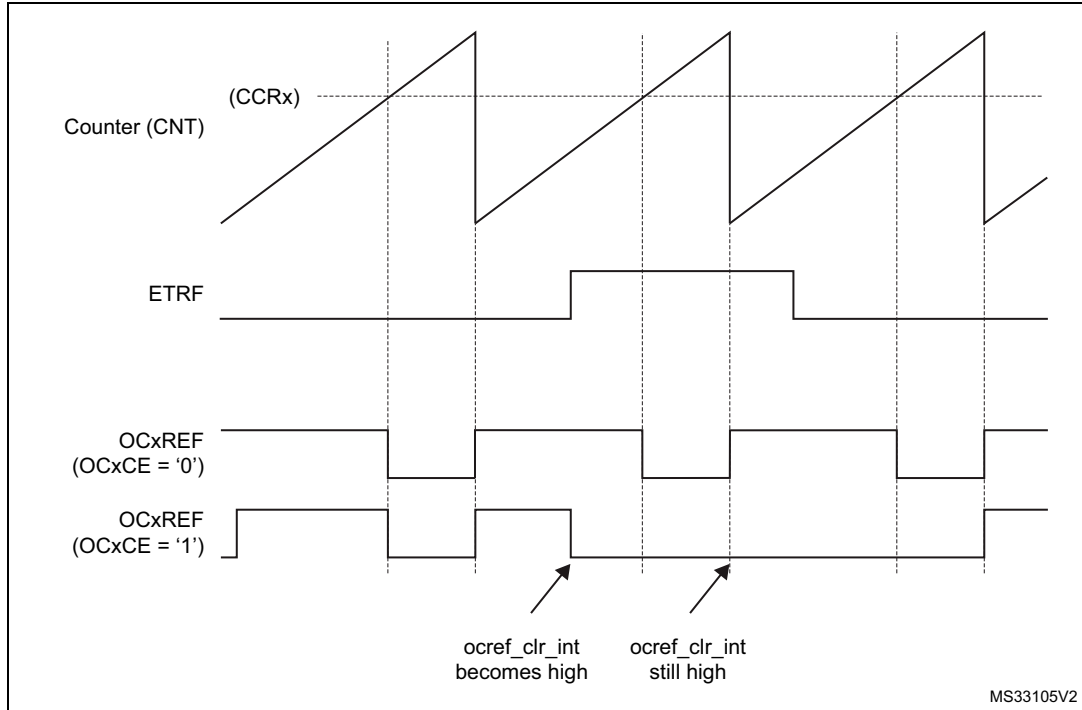
For example, the OCxREF signal can be connected to the output of a comparator to be used for current handling. In this case, ETR must be configured as follows:

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 350 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 350. Clearing TIMx OCxREF



Note: In case of a PWM with a 100% duty cycle (if CCR<sub>x</sub>>ARR), OCxREF is enabled again at the next counter overflow.

### 35.3.13 One-pulse mode

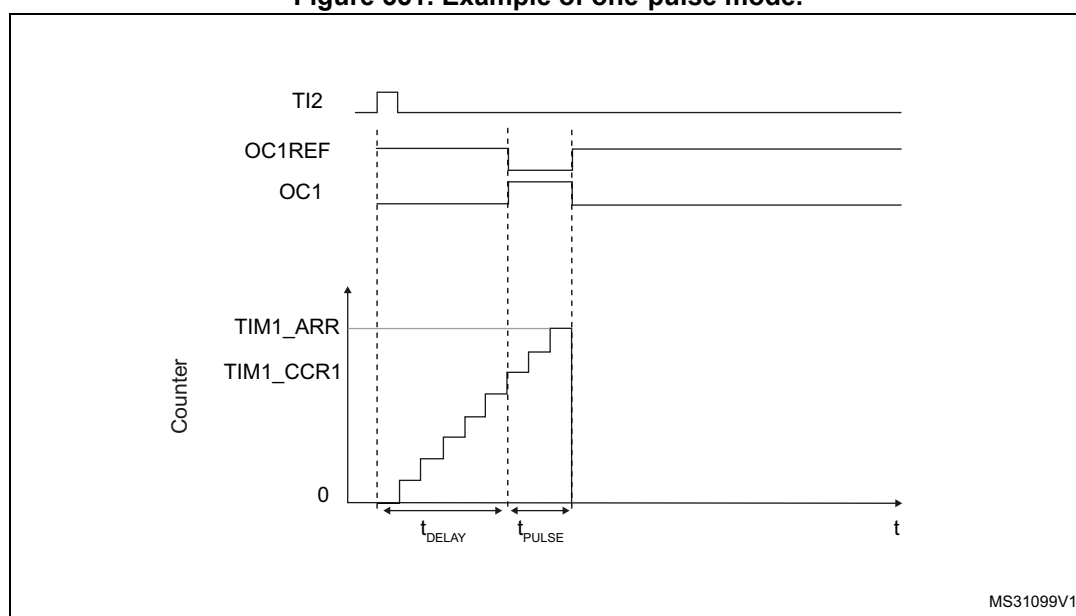
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx\_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- $CNT < CCRx \leq ARR$  (in particular,  $0 < CCRx$ ),

Figure 351. Example of one-pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of  $t_{PULSE}$  and after a delay of  $t_{DELAY}$  as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Map TI2FP2 on TI2 by writing  $CC2S=01$  in the TIMx\_CCMR1 register.
2. TI2FP2 must detect a rising edge, write  $CC2P=0$  and  $CC2NP='0'$  in the TIMx\_CCER register.
3. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing  $TS=110$  in the TIMx\_SMCR register.
4. TI2FP2 is used to start the counter by writing  $SMS$  to '110 in the TIMx\_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The  $t_{\text{DELAY}}$  is defined by the value written in the TIMx\_CCR1 register.
- The  $t_{\text{PULSE}}$  is defined by the difference between the auto-reload value and the compare value (TIMx\_ARR - TIMx\_CCR1).
- Let's say you want to build a waveform with a transition from '0 to '1 when a compare match occurs and a transition from '1 to '0 when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M=111 in the TIMx\_CCMR1 register. You can optionally enable the preload registers by writing OC1PE=1 in the TIMx\_CCMR1 register and ARPE in the TIMx\_CR1 register. In this case you have to write the compare value in the TIMx\_CCR1 register, the auto-reload value in the TIMx\_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0 in this example.

In our example, the DIR and CMS bits in the TIMx\_CR1 register should be low.

You only want 1 pulse (Single mode), so you write '1 in the OPM bit in the TIMx\_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx\_CR1 register is set to '0', so the Repetitive Mode is selected.

#### Particular case: OCx fast enable:

In One-pulse mode, the edge detection on Tlx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay  $t_{\text{DELAY min}}$  we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx\_CCMRx register. Then OCxRef (and OCx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 35.3.14 Retriggerable one pulse mode (OPM)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in [Section 35.3.13](#):

- 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.

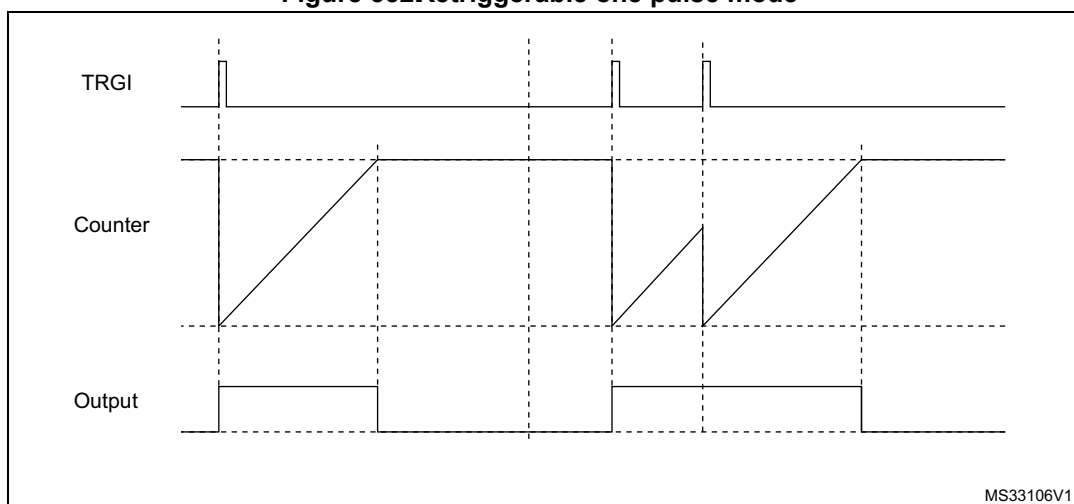
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode CCRx must be above or equal to ARR.

*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 352 Retriggerable one pulse mode



MS33106V1

### 35.3.15 Encoder interface mode

To select Encoder Interface mode write SMS='001 in the TIMx\_SMCR register if the counter is counting on TI2 edges only, SMS=010 if it is counting on TI1 edges only and SMS=011 if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx\_CCER register. CC1NP and CC2NP must be kept cleared. When needed, you can program the input filter as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to [Table 221](#). The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx\_CR1 register written to '1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx\_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx\_ARR register (0 to ARR or ARR down to 0 depending on the direction). So you must configure TIMx\_ARR before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 don't switch at the same time.

**Table 221. 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 353 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 353. Example of counter operation in encoder interface mode**

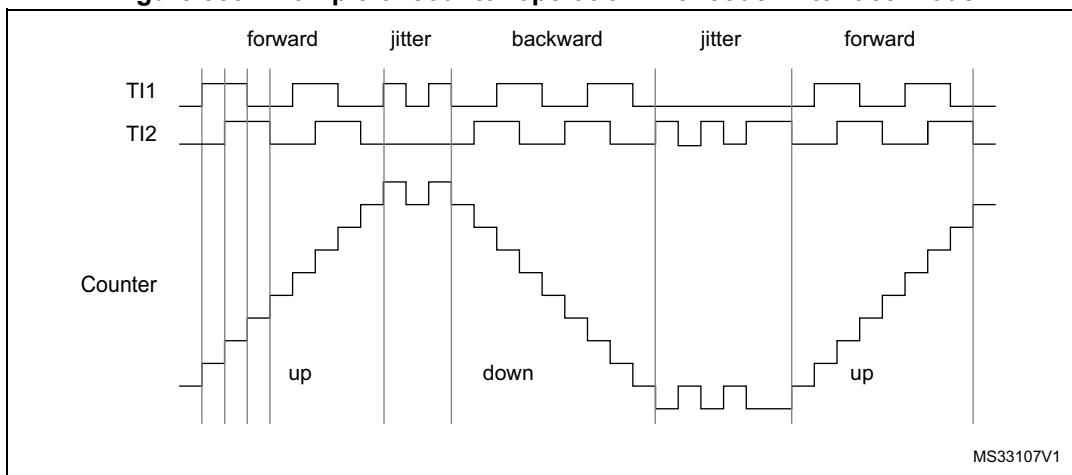
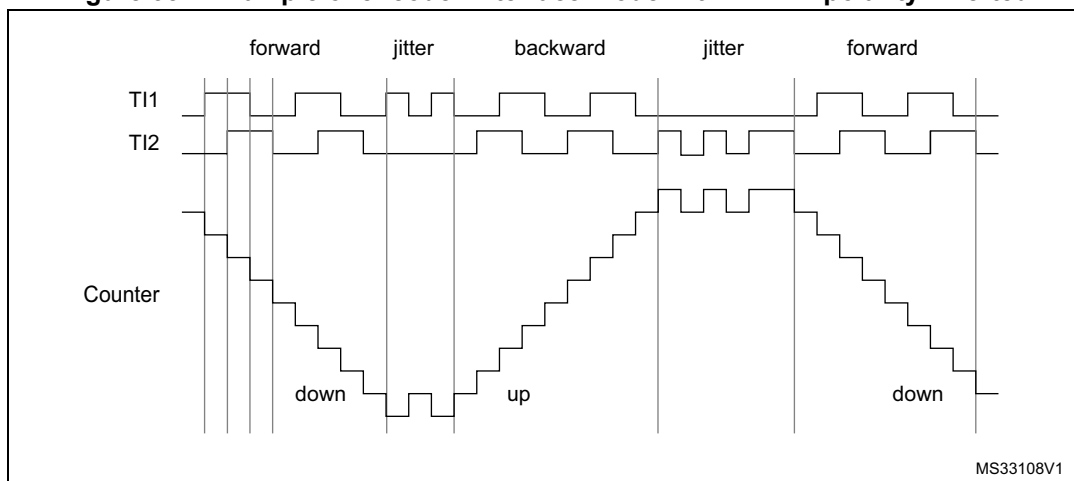


Figure 354 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

**Figure 354. Example of encoder interface mode with TI1FP1 polarity inverted**



The timer, when configured in Encoder Interface mode provides information on the sensor’s current position. You can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. You can do this by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request generated by a Real-Time clock.

### 35.3.16 UIF bit remapping

The IUFREMAP bit in the TIMx\_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register’s bit 31 (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter’s most significant bit is only accessible in write mode).

### 35.3.17 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 34.3.25: Interfacing with Hall sensors on page 1134](#).

### 35.3.18 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

#### Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx\_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx\_ARR, TIMx\_CCRx) are updated.

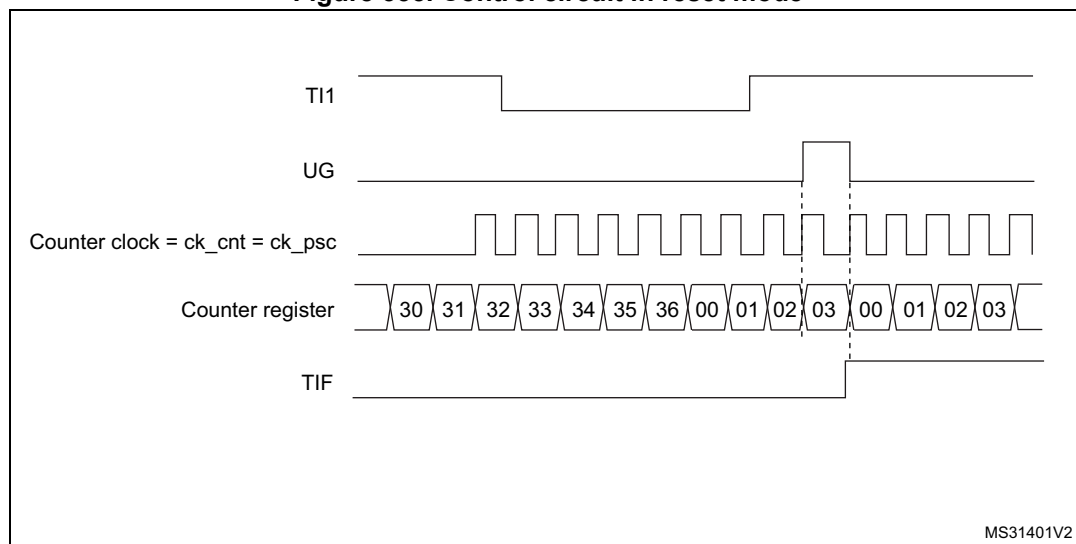
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx\_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx\_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx\_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx\_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx\_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx\_DIER register).

The following figure shows this behavior when the auto-reload register TIMx\_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 355. Control circuit in reset mode



#### Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

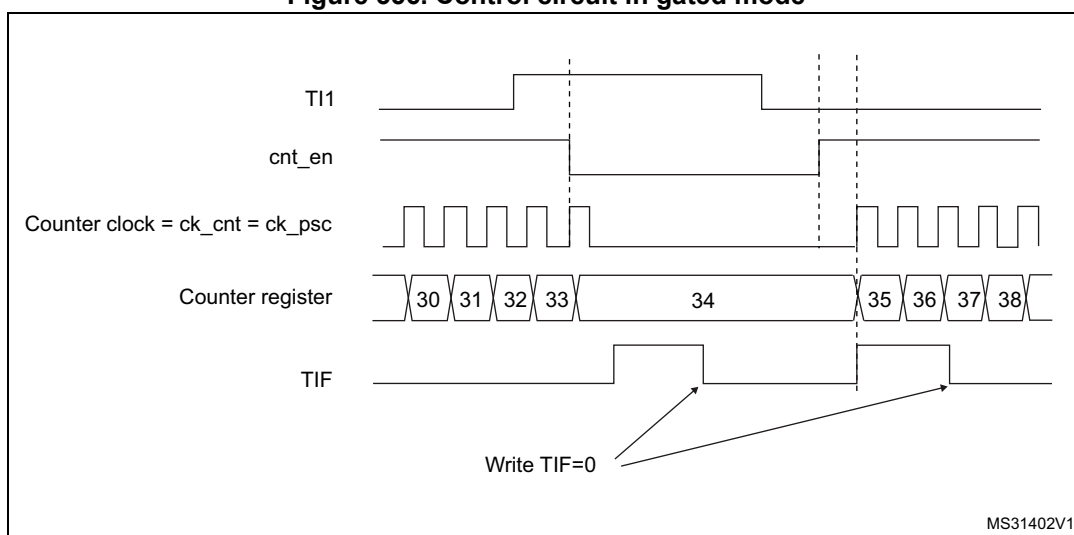
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S=01 in TIMx\_CCMR1 register. Write CC1P=1 and CC1NP=0 in TIMx\_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx\_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx\_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx\_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

**Figure 356. Control circuit in gated mode**



1. The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

**Note:** *The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.*

**Slave mode: Trigger mode**

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx\_CCMR1 register. Write



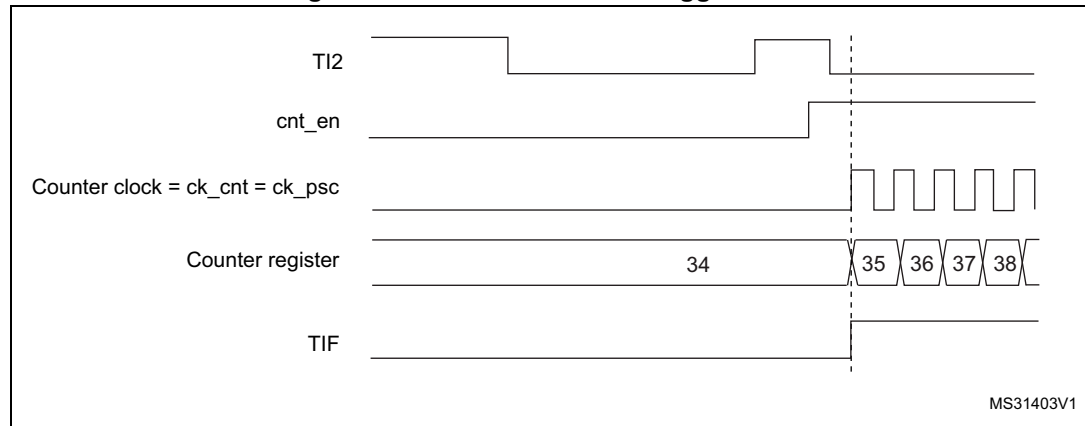
CC2P=1 and CC2NP=0 in TIMx\_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in trigger mode by writing SMS=110 in TIMx\_SMCR register. Select TI2 as the input source by writing TS=110 in TIMx\_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

**Figure 357. Control circuit in trigger mode**



### Slave mode: External Clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select ETR as TRGI through the TS bits of TIMx\_SMCR register.

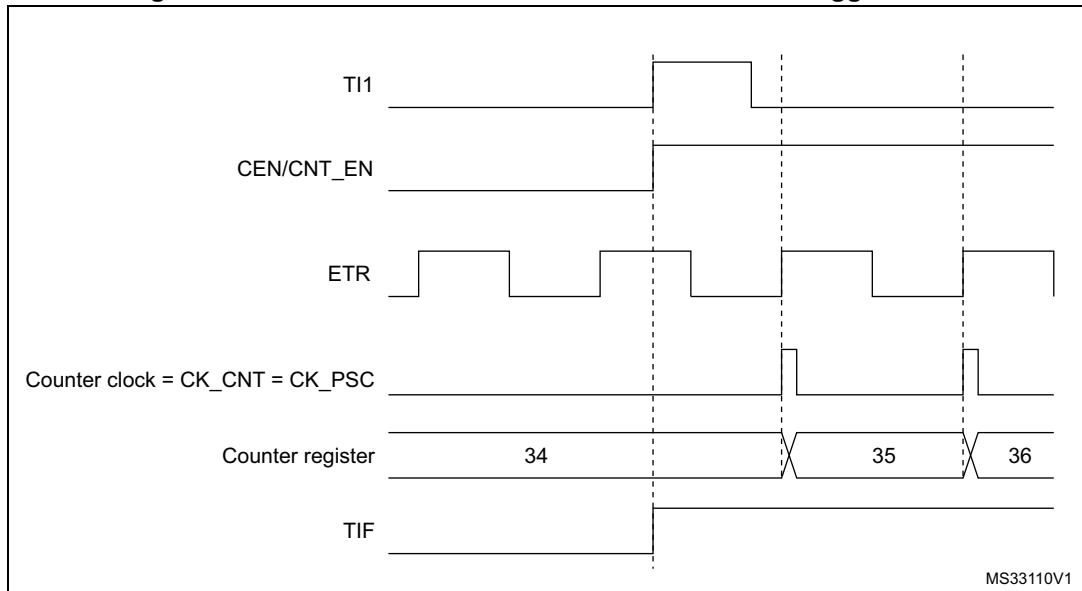
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx\_SMCR register as follows:
  - ETF = 0000: no filter
  - ETPS=00: prescaler disabled
  - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI1:
  - IC1F=0000: no filter.
  - The capture prescaler is not used for triggering and does not need to be configured.
  - CC1S=01 in TIMx\_CCMR1 register to select only the input capture source
  - CC1P=0 and CC1NP=0 in TIMx\_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx\_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

**Figure 358. Control circuit in external clock mode 2 + trigger mode**

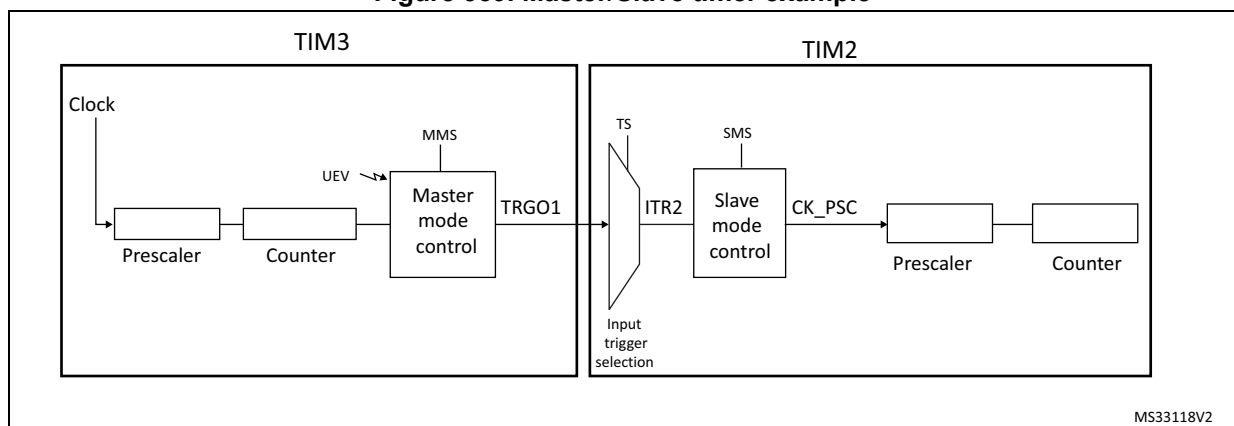


### 35.3.19 Timer synchronization

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 359: Master/Slave timer example* presents an overview of the trigger selection and the master mode selection blocks.

**Figure 359. Master/Slave timer example**



### Using one timer as prescaler for another timer

For example, you can configure TIM3 to act as a prescaler for TIM2. Refer to [Figure 359](#). To do this:

1. Configure TIM3 in master mode so that it outputs a periodic trigger signal on each update event UEV. If you write MMS=010 in the TIM3\_CR2 register, a rising edge is output on TRGO each time an update event is generated.
2. To connect the TRGO output of TIM3 to TIM2, TIM2 must be configured in slave mode using ITR2 as internal trigger. You select this through the TS bits in the TIM2\_SMCR register (writing TS=010).
3. Then you put the slave mode controller in external clock mode 1 (write SMS=111 in the TIM2\_SMCR register). This causes TIM2 to be clocked by the rising edge of the periodic TIM3 trigger signal (which correspond to the TIM3 counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx\_CR1 register).

*Note:* If OCx is selected on TIM3 as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM2.

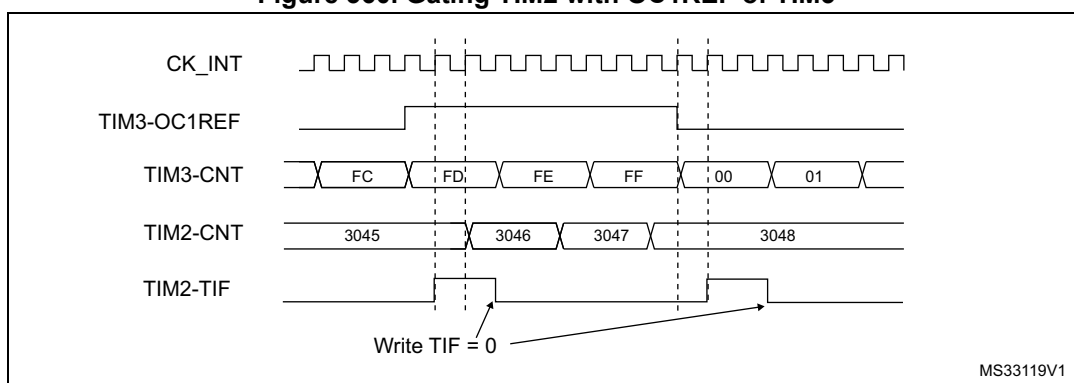
### Using one timer to enable another timer

In this example, we control the enable of TIM2 with the output compare 1 of Timer 3. Refer to [Figure 359](#) for connections. TIM2 counts on the divided internal clock only when OC1REF of TIM3 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK\_INT ( $f_{CK\_CNT} = f_{CK\_INT}/3$ ).

1. Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3\_CR2 register).
2. Configure the TIM3 OC1REF waveform (TIM3\_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM3 (TS=010 in the TIM2\_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2\_SMCR register).
5. Enable TIM2 by writing '1 in the CEN bit (TIM2\_CR1 register).
6. Start TIM3 by writing '1 in the CEN bit (TIM3\_CR1 register).

*Note:* The counter 2 clock is not synchronized with counter 1, this mode only affects the TIM2 counter enable signal.

**Figure 360. Gating TIM2 with OC1REF of TIM3**

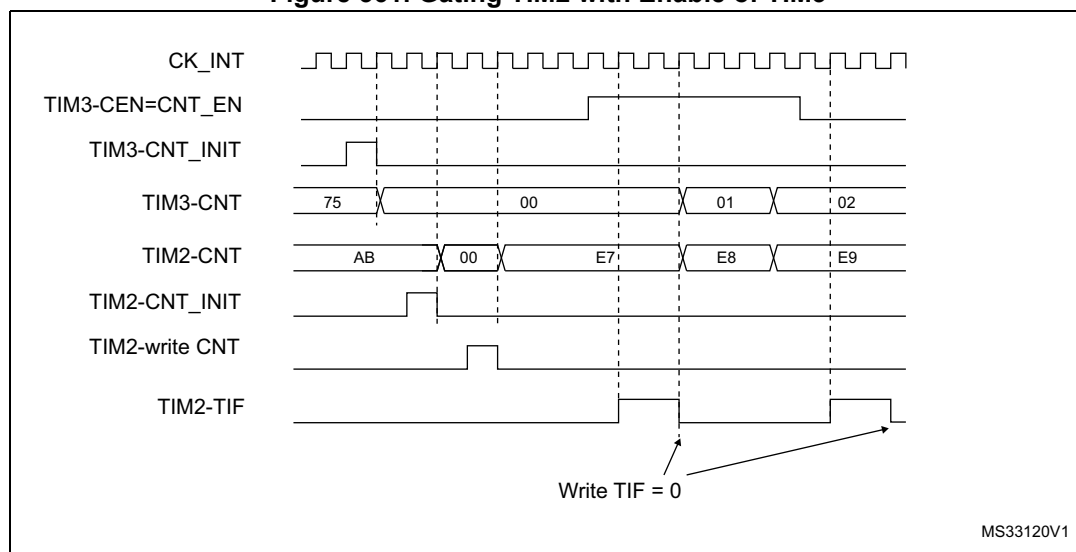


In the example in [Figure 360](#), the TIM2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM3. You can then write any value you want in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx\_EGR registers.

In the next example (refer to [Figure 361](#)), we synchronize TIM3 and TIM2. TIM3 is the master and starts from 0. TIM2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM2 stops when TIM3 is disabled by writing '0 to the CEN bit in the TIM3\_CR1 register:

1. Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3\_CR2 register).
2. Configure the TIM3 OC1REF waveform (TIM3\_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM3 (TS=010 in the TIM2\_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2\_SMCR register).
5. Reset TIM3 by writing '1 in UG bit (TIM3\_EGR register).
6. Reset TIM2 by writing '1 in UG bit (TIM2\_EGR register).
7. Initialize TIM2 to 0xE7 by writing '0xE7' in the TIM2 counter (TIM2\_CNT).
8. Enable TIM2 by writing '1 in the CEN bit (TIM2\_CR1 register).
9. Start TIM3 by writing '1 in the CEN bit (TIM3\_CR1 register).
10. Stop TIM3 by writing '0 in the CEN bit (TIM3\_CR1 register).

**Figure 361. Gating TIM2 with Enable of TIM3**

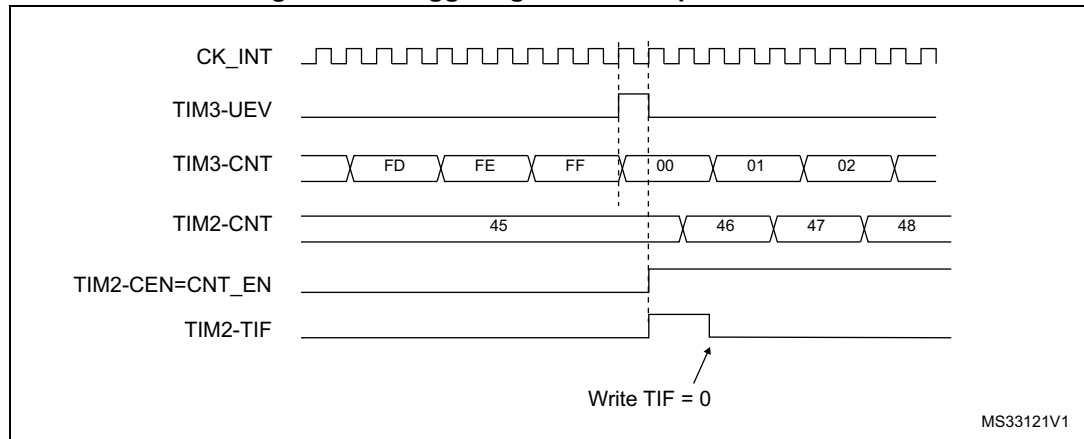


**Using one timer to start another timer**

In this example, we set the enable of Timer 2 with the update event of Timer 3. Refer to [Figure 359](#) for connections. Timer 2 starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by Timer 1. When Timer 2 receives the trigger signal its CEN bit is automatically set and the counter counts until we write '0 to the CEN bit in the TIM2\_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK\_INT ( $f_{CK\_CNT} = f_{CK\_INT}/3$ ).

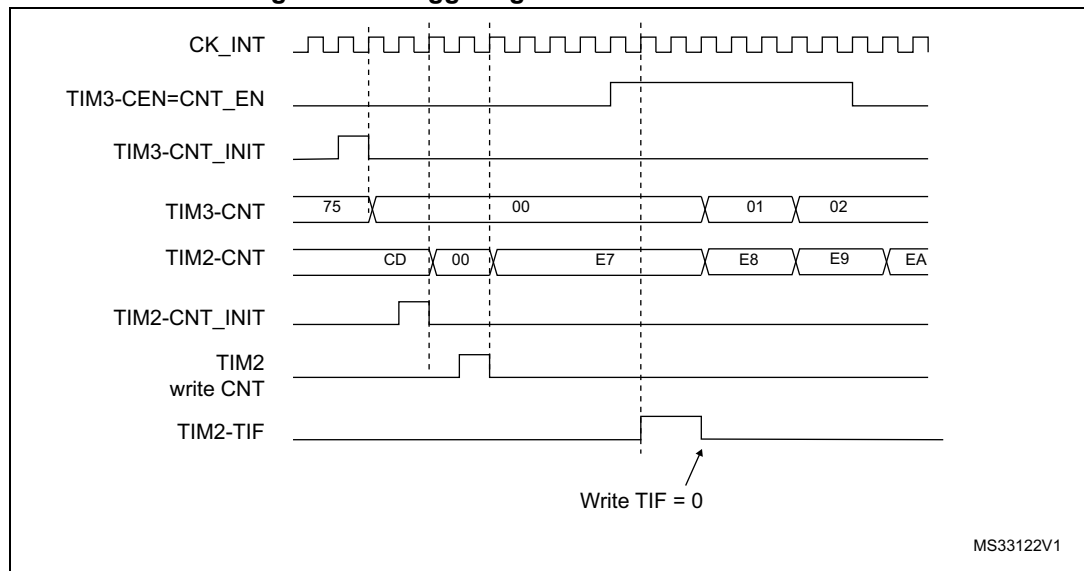
1. Configure TIM3 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM3\_CR2 register).
2. Configure the TIM3 period (TIM3\_ARR registers).
3. Configure TIM2 to get the input trigger from TIM3 (TS=010 in the TIM2\_SMCR register).
4. Configure TIM2 in trigger mode (SMS=110 in TIM2\_SMCR register).
5. Start TIM3 by writing '1 in the CEN bit (TIM3\_CR1 register).

**Figure 362. Triggering TIM2 with update of TIM3**



As in the previous example, you can initialize both counters before starting counting. [Figure 363](#) shows the behavior with the same configuration as in [Figure 362](#) but in trigger mode instead of gated mode (SMS=110 in the TIM2\_SMCR register).

**Figure 363. Triggering TIM2 with Enable of TIM3**



**Starting 2 timers synchronously in response to an external trigger**

In this example, we set the enable of TIM3 when its TI1 input rises, and the enable of TIM2 with the enable of TIM3. Refer to [Figure 359](#) for connections. To ensure the counters are

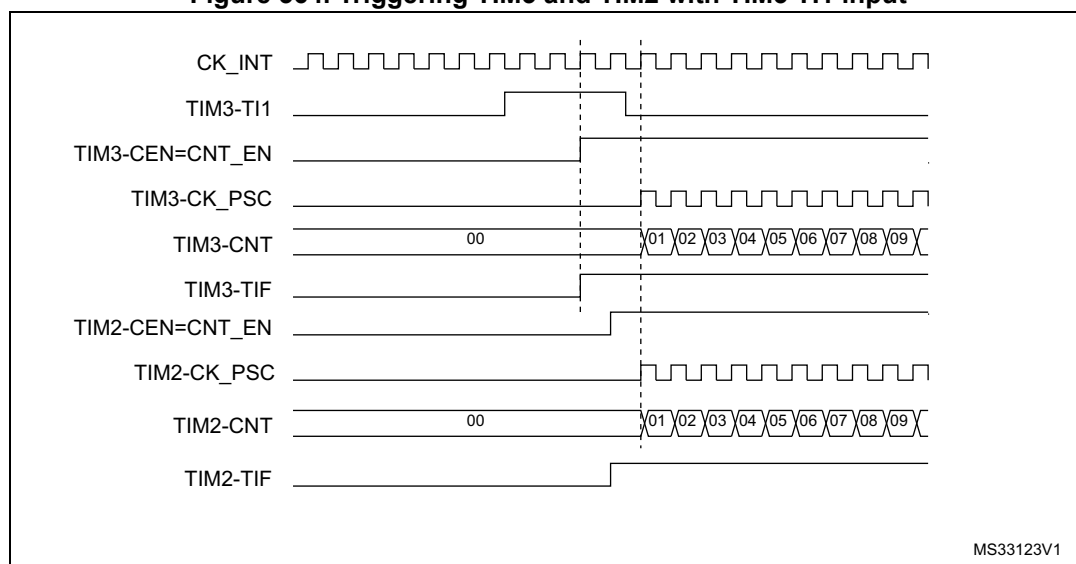
aligned, TIM3 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to TIM2):

1. Configure TIM3 master mode to send its Enable as trigger output (MMS=001 in the TIM3\_CR2 register).
2. Configure TIM3 slave mode to get the input trigger from TI1 (TS=100 in the TIM3\_SMCR register).
3. Configure TIM3 in trigger mode (SMS=110 in the TIM3\_SMCR register).
4. Configure the TIM3 in Master/Slave mode by writing MSM=1 (TIM3\_SMCR register).
5. Configure TIM2 to get the input trigger from TIM3 (TS=000 in the TIM2\_SMCR register).
6. Configure TIM2 in trigger mode (SMS=110 in the TIM2\_SMCR register).

When a rising edge occurs on TI1 (TIM3), both counters starts counting synchronously on the internal clock and both TIF flags are set.

*Note:* In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but you can easily insert an offset between them by writing any of the counter registers (TIMx\_CNT). You can see that the master/slave mode insert a delay between CNT\_EN and CK\_PSC on TIM3.

**Figure 364. Triggering TIM3 and TIM2 with TIM3 TI1 input**



*Note:* The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

### 35.3.20 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.

### 35.3.21 Debug mode

When the microcontroller enters debug mode (Cortex<sup>®</sup>-M4 core - halted), the TIMx counter either continues to work normally or stops, depending on DBG\_TIMx\_STOP configuration bit in DBGMCU module. For more details, refer to [Section 46.16.2: Debug support for timers, RTC, watchdog, bxCAN and I2C](#).

### 35.4 TIM2/TIM3/TIM4/TIM5 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).

#### 35.4.1 TIMx control register 1 (TIMx\_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
				rW		rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

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

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx\_CNT register bit 31.

1: Remapping enabled. UIF status bit is copied to TIMx\_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[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**: 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.

### 35.4.2 TIMx control register 2 (TIMx\_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

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

Bit 7 **TI1S**: TI1 selection

0: The TIMx\_CH1 pin is connected to TI1 input

1: The TIMx\_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

See also [Section 34.3.25: Interfacing with Hall sensors on page 1134](#)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx\_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter enable signal, CNT\_EN, is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx\_SMCR register).

010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred.

(TRGO)

100: **Compare** - OC1REF signal is used as trigger output (TRGO)

101: **Compare** - OC2REF signal is used as trigger output (TRGO)

110: **Compare** - OC3REF signal is used as trigger output (TRGO)

111: **Compare** - OC4REF signal is used as trigger output (TRGO)

*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.

### 35.4.3 TIMx slave mode control register (TIMx\_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

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

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

Bit 16 **SMS[3]**: Slave mode selection - bit 3  
Refer to SMS description - bits 2:0.

Bit 15 **ETP**: External trigger polarity  
This bit selects whether ETR or  $\overline{ETR}$  is used for trigger operations  
0: ETR is non-inverted, active at high level or rising edge  
1: ETR is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable  
This bit enables External clock mode 2.  
0: External clock mode 2 disabled  
1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.  
**1**: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).  
**2**: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).  
**3**: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler  
External trigger signal ETRP frequency must be at most 1/4 of CK\_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.  
00: Prescaler OFF  
01: ETRP frequency divided by 2  
10: ETRP frequency divided by 4  
11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N 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 6:4 **TS**: Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

- 000: Internal Trigger 0 (ITR0).
- 001: Internal Trigger 1 (ITR1).
- 010: Internal Trigger 2 (ITR2).
- 011: Internal Trigger 3 (ITR3).
- 100: TI1 Edge Detector (TI1F\_ED)
- 101: Filtered Timer Input 1 (TI1FP1)
- 110: Filtered Timer Input 2 (TI2FP2)
- 111: External Trigger input (ETRF)

See [Table 222: TIMx internal trigger connection on page 1239](#) for more details on ITRx meaning for each Timer.

*Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.*

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (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=100). 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 222. TIMx internal trigger connection**

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM2	TIM1	TIM8/OTG_FS_SOF <sup>(1)</sup>	TIM3	TIM4
TIM3	TIM1	TIM2	TIM5	TIM4
TIM4	TIM1	TIM2	TIM3	TIM8
TIM5	TIM2	TIM3	TIM4	TIM8

1. Depends on the bit ITR1\_RMP in TIM2\_OR1 register.

### 35.4.4 TIMx DMA/Interrupt enable register (TIMx\_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

- Bit 15 Reserved, must be kept at reset value.
- Bit 14 **TDE**: Trigger DMA request enable  
0: Trigger DMA request disabled.  
1: Trigger DMA request enabled.
- Bit 13 Reserved, must be kept at reset value.
- Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable  
0: CC4 DMA request disabled.  
1: CC4 DMA request enabled.
- Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable  
0: CC3 DMA request disabled.  
1: CC3 DMA request enabled.
- Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable  
0: CC2 DMA request disabled.  
1: CC2 DMA request enabled.
- Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable  
0: CC1 DMA request disabled.  
1: CC1 DMA request enabled.
- Bit 8 **UDE**: Update DMA request enable  
0: Update DMA request disabled.  
1: Update DMA request enabled.
- Bit 7 Reserved, must be kept at reset value.
- Bit 6 **TIE**: Trigger interrupt enable  
0: Trigger interrupt disabled.  
1: Trigger interrupt enabled.
- Bit 5 Reserved, must be kept at reset value.
- Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable  
0: CC4 interrupt disabled.  
1: CC4 interrupt enabled.
- Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable  
0: CC3 interrupt disabled.  
1: CC3 interrupt enabled.

- Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable  
 0: CC2 interrupt disabled.  
 1: CC2 interrupt enabled.
- Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable  
 0: CC1 interrupt disabled.  
 1: CC1 interrupt enabled.
- Bit 0 **UIE**: Update interrupt enable  
 0: Update interrupt disabled.  
 1: Update interrupt enabled.

### 35.4.5 TIMx status register (TIMx\_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	CC4OF	CC3OF	CC2OF	CC1OF	Res	Res	TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag  
 refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag  
 refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag  
 refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag  
 This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.  
 0: No overcapture has been detected.  
 1: The counter value has been captured in TIMx\_CCR1 register while CC1IF flag was already set

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

Bit 6 **TIF**: Trigger interrupt flag  
 This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.  
 0: No trigger event occurred.  
 1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag  
 Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag  
 Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag  
 Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag  
**If channel CC1 is configured as output:** This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx\_CR1 register description) and in retriggerable one pulse mode. It is cleared by software.  
 0: No match.  
 1: The content of the counter TIMx\_CNT has matched the content of the TIMx\_CCR1 register.  
**If channel CC1 is configured as input:** This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx\_CCR1 register.  
 0: No input capture occurred.  
 1: The counter value has been captured in TIMx\_CCR1 register (An edge has been detected on IC1 which matches the selected polarity).

Bit 0 **UIF**: Update interrupt flag  
 This bit is set by hardware on an update event. It is cleared by software.  
 0: No update occurred  
 1: Update interrupt pending. This bit is set by hardware when the registers are updated: At overflow or underflow (for TIM2 to TIM4) and if UDIS=0 in the TIMx\_CR1 register. When CNT is reinitialized by software using the UG bit in TIMx\_EGR register, if URS=0 and UDIS=0 in the TIMx\_CR1 register. When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx\_CR1 register.

### 35.4.6 TIMx event generation register (TIMx\_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

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

Bit 6 **TG**: Trigger generation  
 This bit is set by software in order to generate an event, it is automatically cleared by hardware.  
 0: No action  
 1: The TIF flag is set in TIMx\_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation  
 Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation  
 Refer to CC1G description



- Bit 2 **CC2G**: Capture/compare 2 generation  
Refer to CC1G description
- Bit 1 **CC1G**: Capture/compare 1 generation  
This bit is set by software in order to generate an event, it is automatically cleared by hardware.  
0: No action  
1: A capture/compare event is generated on channel 1:  
**If channel CC1 is configured as output:**  
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.  
**If channel CC1 is configured as input:**  
The current value of the counter is captured in TIMx\_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.
- Bit 0 **UG**: Update generation  
This bit can be set by software, it is automatically cleared by hardware.  
0: No action  
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx\_ARR) if DIR=1 (downcounting).

### 35.4.7 TIMx capture/compare mode register 1 (TIMx\_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
							Res.								Res.
							rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

#### Output compare mode

- Bits 31:25 Reserved, always read as 0.
- Bit 24 **OC2M[3]**: Output Compare 2 mode - bit 3
- Bits 23:17 Reserved, always read as 0.
- Bit 16 **OC1M[3]**: Output Compare 1 mode - bit 3



- Bit 15 **OC2CE**: Output compare 2 clear enable
- Bits 14:12 **OC2M[2:0]**: Output compare 2 mode  
refer to OC1M description on bits 6:4
- Bit 11 **OC2PE**: Output compare 2 preload enable
- Bit 10 **OC2FE**: Output compare 2 fast enable
- Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection  
This bit-field defines the direction of the channel (input/output) as well as the used input.  
00: CC2 channel is configured as output  
01: CC2 channel is configured as input, IC2 is mapped on TI2  
10: CC2 channel is configured as input, IC2 is mapped on TI1  
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx\_SMCR register)  
*Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx\_CCER).*
- Bit 7 **OC1CE**: Output compare 1 clear enable  
0: OC1Ref is not affected by the ETRF input  
1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 6:4 **OC1M**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx\_CCR1 and the counter TIMx\_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0011: Toggle - OC1REF toggles when TIMx\_CNT=TIMx\_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx\_CNT<TIMx\_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF=0) as long as TIMx\_CNT>TIMx\_CCR1 else active (OC1REF=1).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx\_CNT<TIMx\_CCR1 else active. In downcounting, channel 1 is active as long as TIMx\_CNT>TIMx\_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

*Note: 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S=00 (the channel is configured in output).*

*2: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.*

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx\_CCR1 disabled. TIMx\_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx\_CCR1 enabled. Read/Write operations access the preload register. TIMx\_CCR1 preload value is loaded in the active register at each update event.

*Note:* **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S=00 (the channel is configured in output).

**2:** The PWM mode can be used without validating the preload register only in one-pulse mode (OPM bit set in TIMx\_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, IC1 is mapped on TI1.

10: CC1 channel is configured as input, IC1 is mapped on TI2.

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note:* CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx\_CCER).

**Input capture mode**

Bits 31:16 Reserved, always read as 0.

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, IC2 is mapped on TI2.

10: CC2 channel is configured as input, IC2 is mapped on TI1.

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note:* CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx\_CCER).

Bits 7:4 **IC1F**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N 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**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E=0 (TIMx\_CCER register).

- 00: no prescaler, capture is done each time an edge is detected on the capture input
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, IC1 is mapped on TI1
- 10: CC1 channel is configured as input, IC1 is mapped on TI2
- 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx\_CCER).*

### 35.4.8 TIMx capture/compare mode register 2 (TIMx\_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M [3]
							Res.								Res.
							rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw



## Output compare mode

Bits 31:25 Reserved, always read as 0.

Bit 24 **OC4M[3]**: Output Compare 2 mode - bit 3

Bits 23:17 Reserved, always read as 0.

Bit 16 **OC3M[3]**: Output Compare 1 mode - bit 3

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 14:12 **OC4M**: Output compare 4 mode

Refer to OC1M description (bits 6:4 in TIMx\_CCMR1 register)

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx\_CCER).*

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 6:4 **OC3M**: Output compare 3 mode

Refer to OC1M description (bits 6:4 in TIMx\_CCMR1 register)

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx\_CCER).*

## Input capture mode

Bits 31:16 Reserved, always read as 0.

Bits 15:12 **IC4F**: Input capture 4 filter

Bits 11:10 **IC4PSC**: Input capture 4 prescaler

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx\_CCER).*

Bits 7:4 **IC3F**: Input capture 3 filter

Bits 3:2 **IC3PSC**: Input capture 3 prescaler

Bits 1:0 **CC3S**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx\_CCER).*

### 35.4.9 TIMx capture/compare enable register (TIMx\_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rW		rW	rW	rW		rW	rW	rW		rW	rW	rW		rW	rW

Bit 15 **CC4NP**: Capture/Compare 4 output Polarity.

Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/Compare 4 output Polarity.

Refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable.

refer to CC1E description

Bit 11 **CC3NP**: Capture/Compare 3 output Polarity.

Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: Capture/Compare 3 output Polarity.

Refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable.

Refer to CC1E description

Bit 7 **CC2NP**: Capture/Compare 2 output Polarity.

Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output Polarity.

refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable.

Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 output Polarity.

**CC1 channel configured as output:** CC1NP must be kept cleared in this case.

**CC1 channel configured as input:** This bit is used in conjunction with CC1P to define T11FP1/TI2FP1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.

**CC1 channel configured as output:**

0: OC1 active high

1: OC1 active low

**CC1 channel configured as input:** CC1NP/CC1P bits select TI1FP1 and TI2FP1 polarity for trigger or capture operations.

00: noninverted/rising edge

Circuit is sensitive to TlxFP1 rising edge (capture, trigger in reset, external clock or trigger mode), TlxFP1 is not inverted (trigger in gated mode, encoder mode).

01: inverted/falling edge

Circuit is sensitive to TlxFP1 falling edge (capture, trigger in reset, external clock or trigger mode), TlxFP1 is inverted (trigger in gated mode, encoder mode).

10: reserved, do not use this configuration.

11: noninverted/both edges

Circuit is sensitive to both TlxFP1 rising and falling edges (capture, trigger in reset, external clock or trigger mode), TlxFP1 is not inverted (trigger in gated mode). This configuration must not be used for encoder mode.

Bit 0 **CC1E**: Capture/Compare 1 output enable.

CC1 channel configured as output:

0: Off - OC1 is not active

1: On - OC1 signal is output on the corresponding output pin

**CC1 channel configured as input:** This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx\_CCR1) or not.

0: Capture disabled

1: Capture enabled

**Table 223. Output control bit for standard OCx channels**

CCxE bit	OCx output state
0	Output Disabled (OCx=0, OCx_EN=0)
1	OCx=OCxREF + Polarity, OCx_EN=1

*Note:* The state of the external IO pins connected to the standard OCx channels depends on the OCx channel state and the GPIO and AFIO registers.

### 35.4.10 TIMx counter (TIMx\_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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



Bit 31 Value depends on UIFREMAP in TIMx\_CR1.

If UIFREMAP = 0

**CNT[31]:** Most significant bit of counter value (on TIM2 and TIM5)

Reserved on other timers

If UIFREMAP = 1

**UIFCPY:** UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx\_ISR register

Bits 30:16 **CNT[30:16]:** Most significant part counter value (on TIM2 and TIM5)

Bits 15:0 **CNT[15:0]:** Least significant part of counter value

### 35.4.11 TIMx prescaler (TIMx\_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]:** Prescaler value

The counter clock frequency CK\_CNT is equal to  $f_{CK\_PSC} / (PSC[15:0] + 1)$ .

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx\_EGR register or through trigger controller when configured in “reset mode”).

### 35.4.12 TIMx auto-reload register (TIMx\_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 31:16 **ARR[31:16]:** High auto-reload value (on TIM2 and TIM5)

Bits 15:0 **ARR[15:0]:** Low Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the [Section 35.3.1: Time-base unit on page 1191](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 35.4.13 TIMx capture/compare register 1 (TIMx\_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] (depending on timers)															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR1[31:16]**: High Capture/Compare 1 value (on TIM2 and TIM5)

Bits 15:0 **CCR1[15:0]**: Low Capture/Compare 1 value

**If channel CC1 is configured as output:**

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signaled on OC1 output.

**If channel CC1 is configured as input:**

CCR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx\_CCR1 register is read-only and cannot be programmed.

### 35.4.14 TIMx capture/compare register 2 (TIMx\_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] (depending on timers)															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR2[31:16]**: High Capture/Compare 2 value (on TIM2 and TIM5)

Bits 15:0 **CCR2[15:0]**: Low Capture/Compare 2 value

**If channel CC2 is configured as output:**

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signalled on OC2 output.

**If channel CC2 is configured as input:**

CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx\_CCR2 register is read-only and cannot be programmed.

### 35.4.15 TIMx capture/compare register 3 (TIMx\_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] (depending on timers)															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:16 **CCR3[31:16]**: High Capture/Compare 3 value (on TIM2 and TIM5)

Bits 15:0 **CCR3[15:0]**: Low Capture/Compare value

**If channel CC3 is configured as output:**

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signalled on OC3 output.

**If channel CC3 is configured as input:**

CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx\_CCR3 register is read-only and cannot be programmed.

### 35.4.16 TIMx capture/compare register 4 (TIMx\_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] (depending on timers)															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:16 **CCR4[31:16]**: High Capture/Compare 4 value (on TIM2 and TIM5)

Bits 15:0 **CCR4[15:0]**: Low Capture/Compare value

- if CC4 channel is configured as output (CC4S bits):  
 CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.  
 The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signalled on OC4 output.
- if CC4 channel is configured as input (CC4S bits in TIMx\_CCMR4 register):  
 CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx\_CCR4 register is read-only and cannot be programmed.

### 35.4.17 TIMx DMA control register (TIMx\_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the number of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address).

- 00000: 1 transfer,
- 00001: 2 transfers,
- 00010: 3 transfers,
- ...
- 10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit vector defines the base-address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register.

Example:

- 00000: TIMx\_CR1
- 00001: TIMx\_CR2
- 00010: TIMx\_SMCR
- ...

**Example:** Let us consider the following transfer: DBL = 7 transfers & DBA = TIMx\_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx\_CR1 address.

### 35.4.18 TIMx DMA address for full transfer (TIMx\_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address  $(\text{TIMx\_CR1 address}) + (\text{DBA} + \text{DMA index}) \times 4$

where TIMx\_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx\_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx\_DCR).

### 35.4.19 TIM2 option register 1 (TIM2\_OR1)

Address offset: 0x50

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI4_RMP[1:0]		ETR1_RMP	ITR1_RMP
													rw	rw	rw

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

Bits 3:2 **TI4\_RMP[1:0]**: Input Capture 4 remap

- 00: TIM2 input capture 4 is connected to I/O
- 01: TIM2 input capture 4 is connected to COMP1\_OUT
- 10: TIM2 input capture 4 is connected to COMP2\_OUT
- 11: TIM2 input capture 4 is connected to logical OR between COMP1\_OUT and COMP2\_OUT

Bit 1 **ETR1\_RMP**: External trigger remap

- 0: TIM2\_ETR is connected to I/O
- 1: TIM2\_ETR is connected to LSE

Bit 0 **ITR1\_RMP**: Internal trigger 1 remap

- 0: TIM2\_ITR1 is connected to TIM8\_TRGO
- 1: TIM2\_ITR1 is connected to OTG\_FS\_SOF

### 35.4.20 TIM3 option register 1 (TIM3\_OR1)

Address offset: 0x50

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **TI1\_RMP[1:0]**: Input Capture 1 remap

- 00: TIM3 input capture 1 is connected to I/O
- 01: TIM3 input capture 1 is connected to COMP1\_OUT
- 10: TIM3 input capture 1 is connected to COMP2\_OUT
- 11: TIM3 input capture 1 is connected to logical OR between COMP1\_OUT and COMP2\_OUT

### 35.4.21 TIM2 option register 2 (TIM2\_OR2)

Address offset: 0x60

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETR SEL2
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

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

Bits 16:14 **ETRSEL[2:0]**: ETR source selection

These bits select the ETR input source.

000: TIM2\_ETR source is selected with the ETR1\_RMP bitfield in TIM2\_OR1 register

001: COMP1 output connected to ETR input

010: COMP2 output connected to ETR input

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 13:0 Reserved, must be kept at reset value.

### 35.4.22 TIM3 option register 2 (TIM3\_OR2)

Address offset: 0x60

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETR SEL2
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

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

Bits 16:14 **ETRSEL[2:0]**: ETR source selection

These bits select the ETR input source.

000: ETR input is connected to I/O

001: COMP1 output connected to ETR input

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 13:0 Reserved, must be kept at reset value.

35.4.23 TIMx register map

TIMx registers are mapped as described in the table below:

Table 224. TIM2/TIM3/TIM4/TIM5 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	TIMx_CR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	UJFREMA	Res	CKD [1:0]	ARPE	CMS [1:0]	DIR	OPM	URS	UDIS	CEN			
	Reset value																						0		0	0	0	0	0	0	0	0		
0x04	TIMx_CR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	T1S	MMS[2:0]		CCDS	Res	Res	Res		
	Reset value																									0	0	0	0	0				
0x08	TIMx_SMCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SMS[3]	ETP	ECE	ETPS [1:0]	ETF[3:0]			MSM	TS[2:0]		SMS[2:0]							
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0C	TIMx_DIER	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res	TIE	Res	CC4IE	CC3IE	CC2IE	CC1IE	UIE	
	Reset value																		0	0	0	0	0	0	0	0	Res	0	Res	0	0	0	0	
0x10	TIMx_SR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																										Res	0	Res	0	0	0	0	0
0x14	TIMx_EGR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TG	Res	CC4G	CC3G	CC2G	CC1G	UG	
	Reset value																										0	Res	0	0	0	0	0	
0x18	TIMx_CCMR1 Output Compare mode	Res	Res	Res	Res	Res	Res	Res	OC2M[3]	Res	Res	Res	Res	Res	Res	Res	Res	OC1M[3]	OC2CE	OC2M [2:0]		OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]		OC1PE	OC1FE	CC1S [1:0]			
	Reset value								0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	TIMx_CCMR1 Input Capture mode	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	IC2F[3:0]			IC2 PSC [1:0]	CC2S [1:0]		IC1F[3:0]		IC1 PSC [1:0]	CC1S [1:0]						
Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x1C	TIMx_CCMR2 Output Compare mode	Res	Res	Res	Res	Res	Res	Res	OC4M[3]	Res	Res	Res	Res	Res	Res	Res	Res	OC3M[3]	OC4CE	OC4M [2:0]		OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]		OC3PE	OC3FE	CC3S [1:0]			
	Reset value								0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	TIMx_CCMR2 Input Capture mode	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	IC4F[3:0]			IC4 PSC [1:0]	CC4S [1:0]		IC3F[3:0]		IC3 PSC [1:0]	CC3S [1:0]						
Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x20	TIMx_CCER	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	







Table 224. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
0x50	TIM3_OR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T11_RMP[1:0]		
	Reset value																																			0	
0x60	TIM2_OR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																0	0	0																		
0x60	TIM3_OR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																0	0	0																		

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.

## 36 General-purpose timers (TIM15/TIM16/TIM17)

### 36.1 TIM15/TIM16/TIM17 introduction

The TIM15/TIM16/TIM17 timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16/TIM17 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in [Section 36.4.22: Timer synchronization \(TIM15\)](#).

### 36.2 TIM15 main features

TIM15 includes the following features:

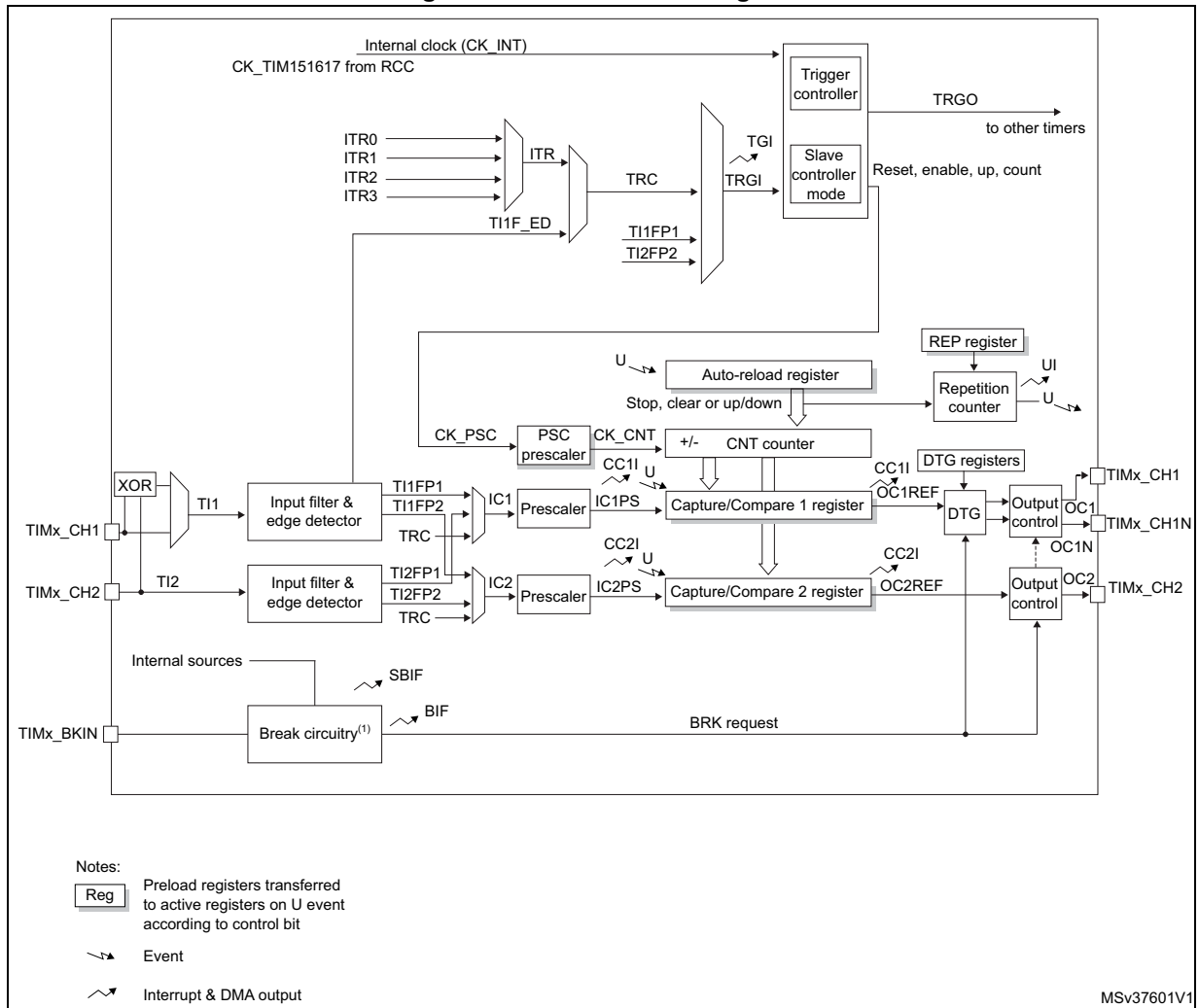
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)

### 36.3 TIM16/TIM17 main features

The TIM16/TIM17 timers include the following features:

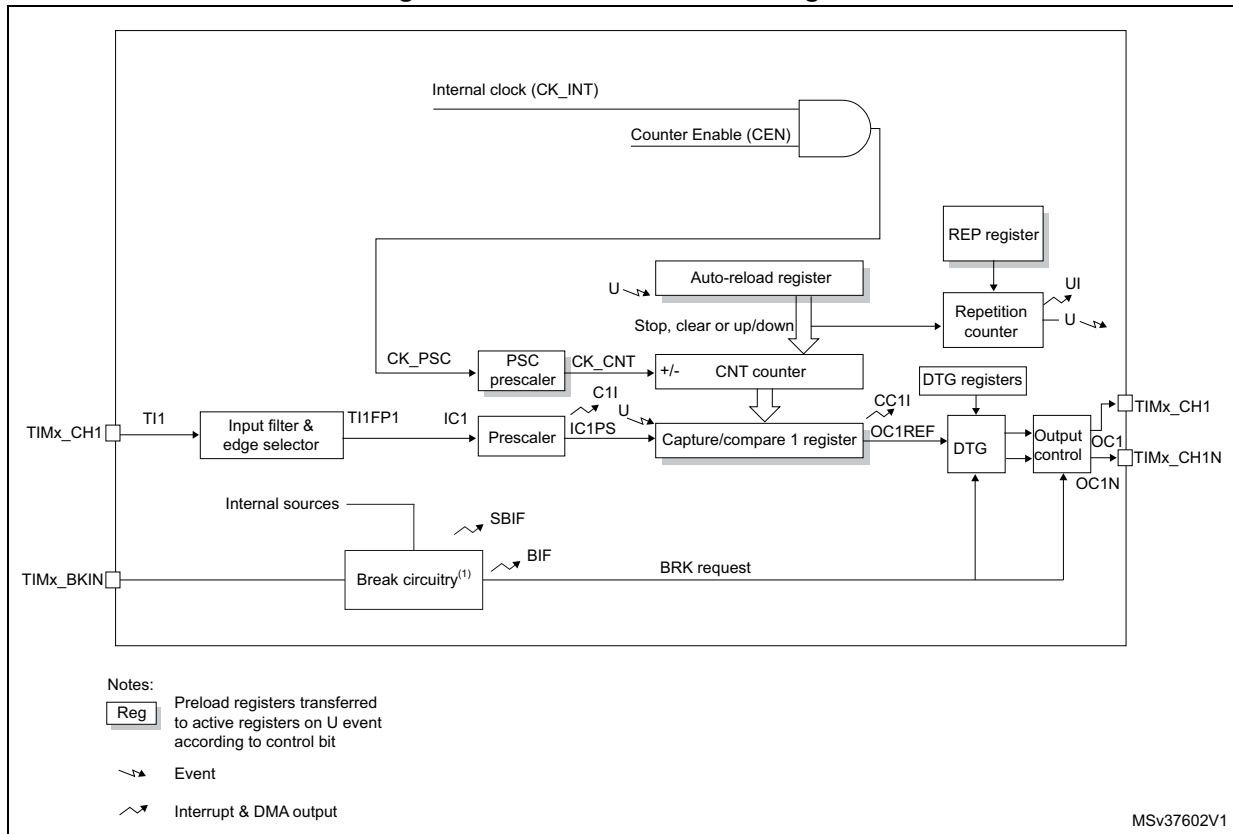
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input

Figure 365. TIM15 block diagram



- The internal break event source can be:
  - A clock failure event generated by CSS. For further information on the CSS, refer to [Section 6.2.10: Clock security system \(CSS\)](#)
  - A PVD output
  - SRAM parity error signal
  - Cortex<sup>®</sup>-M4 LOCKUP (Hardfault) output
  - COMP output

Figure 366. TIM16/TIM17 block diagram



- The internal break event source can be:
  - A clock failure event generated by CSS. For further information on the CSS, refer to [Section 6.2.10: Clock security system \(CSS\)](#)
  - A PVD output
  - SRAM parity error signal
  - Cortex<sup>®</sup>-M4 LOCKUP (Hardfault) output
  - COMP output

## 36.4 TIM15/TIM16/TIM17 functional description

### 36.4.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx\_CNT)
- Prescaler register (TIMx\_PSC)
- Auto-reload register (TIMx\_ARR)
- Repetition counter register (TIMx\_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx\_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx\_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIMx\_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx\_CR1 register.

#### Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx\_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 367* and *Figure 368* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 367. Counter timing diagram with prescaler division change from 1 to 2

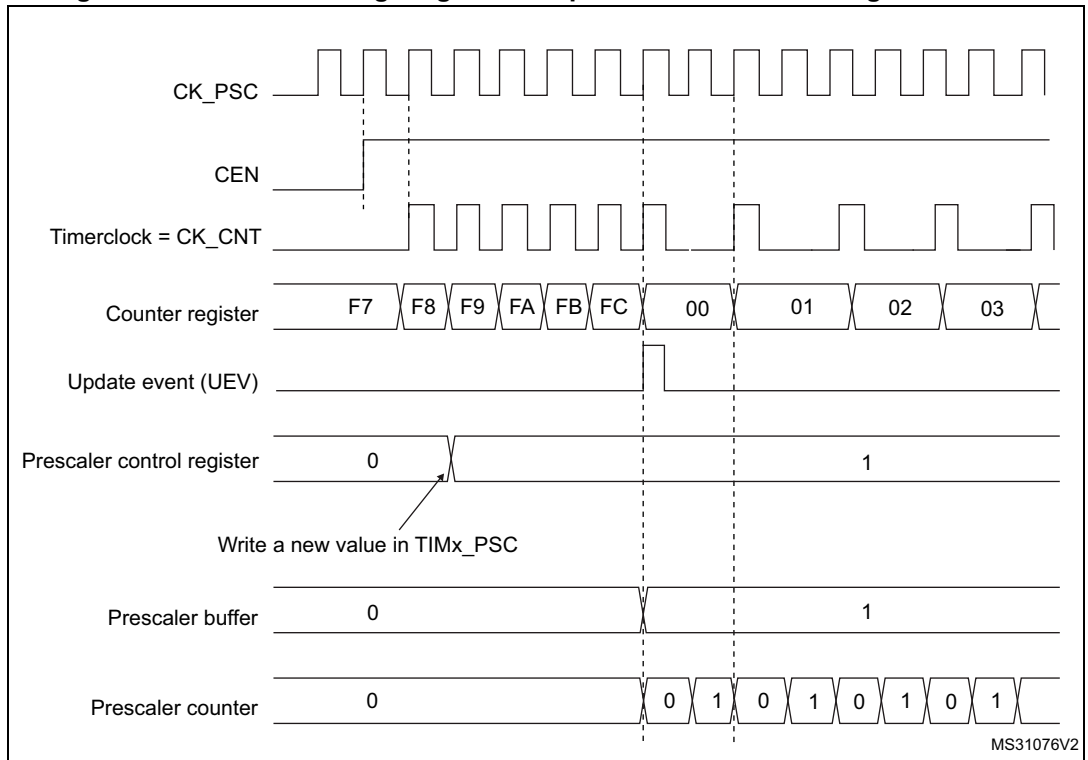
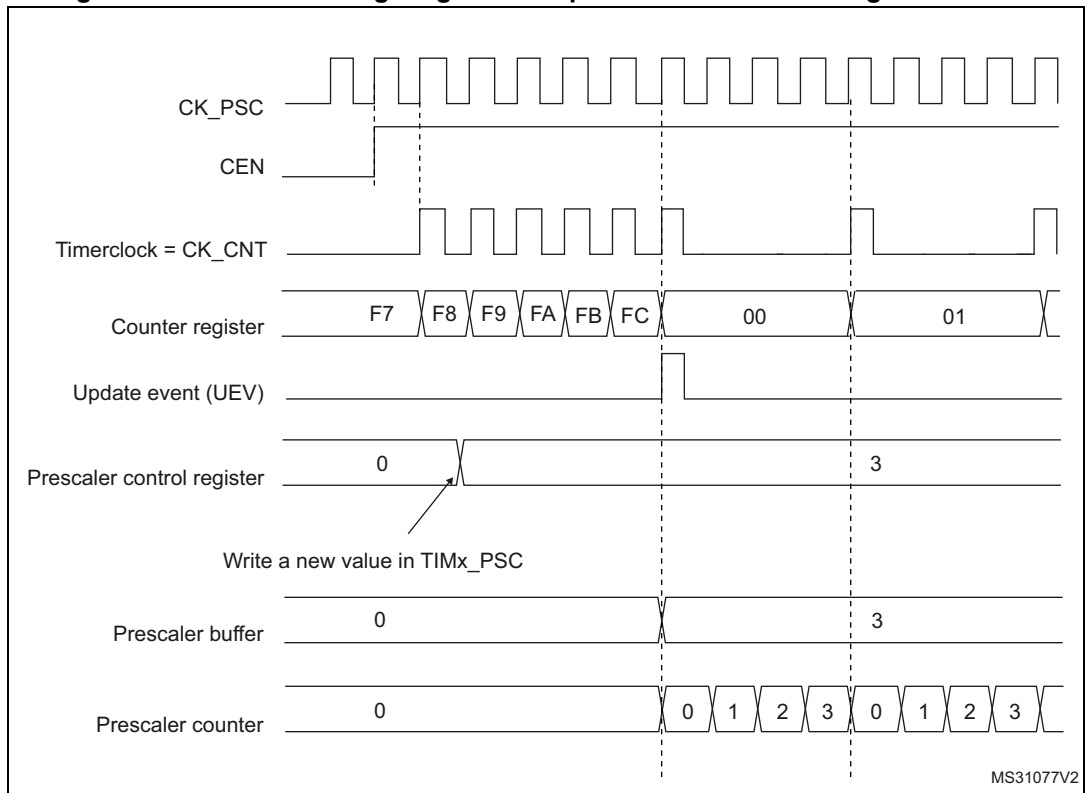


Figure 368. Counter timing diagram with prescaler division change from 1 to 4



## 36.4.2 Counter modes

### Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx\_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

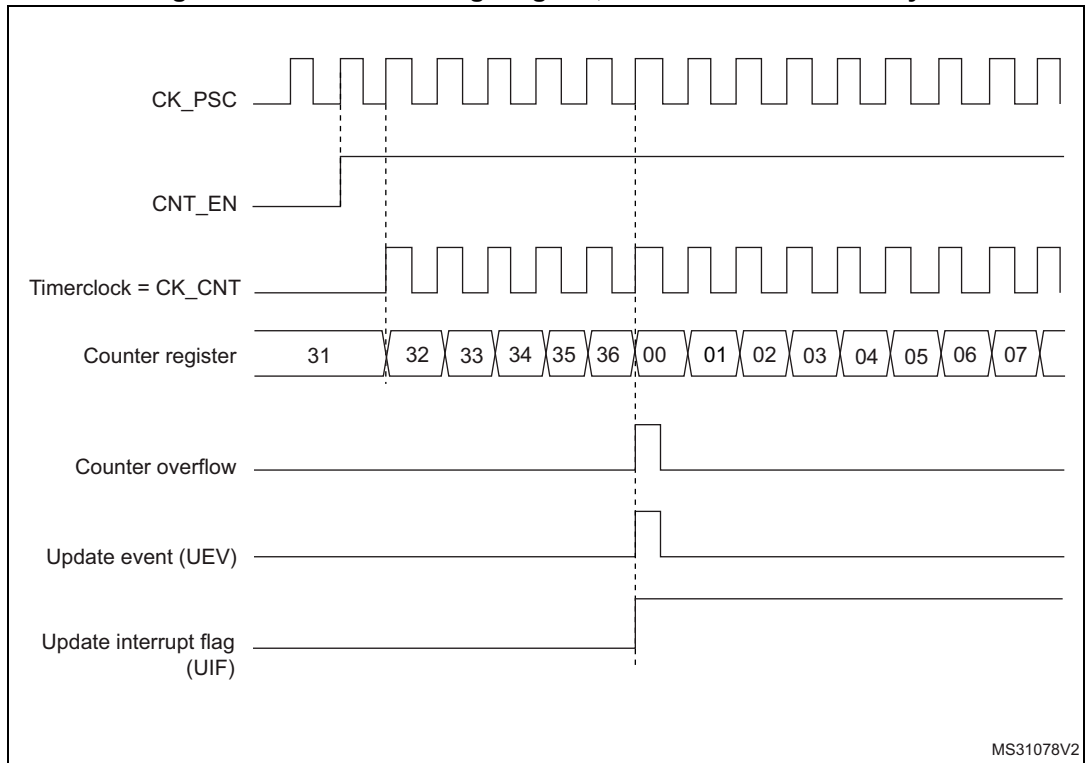
When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx\_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx\_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR=0x36.



**Figure 369. Counter timing diagram, internal clock divided by 1**



**Figure 370. Counter timing diagram, internal clock divided by 2**

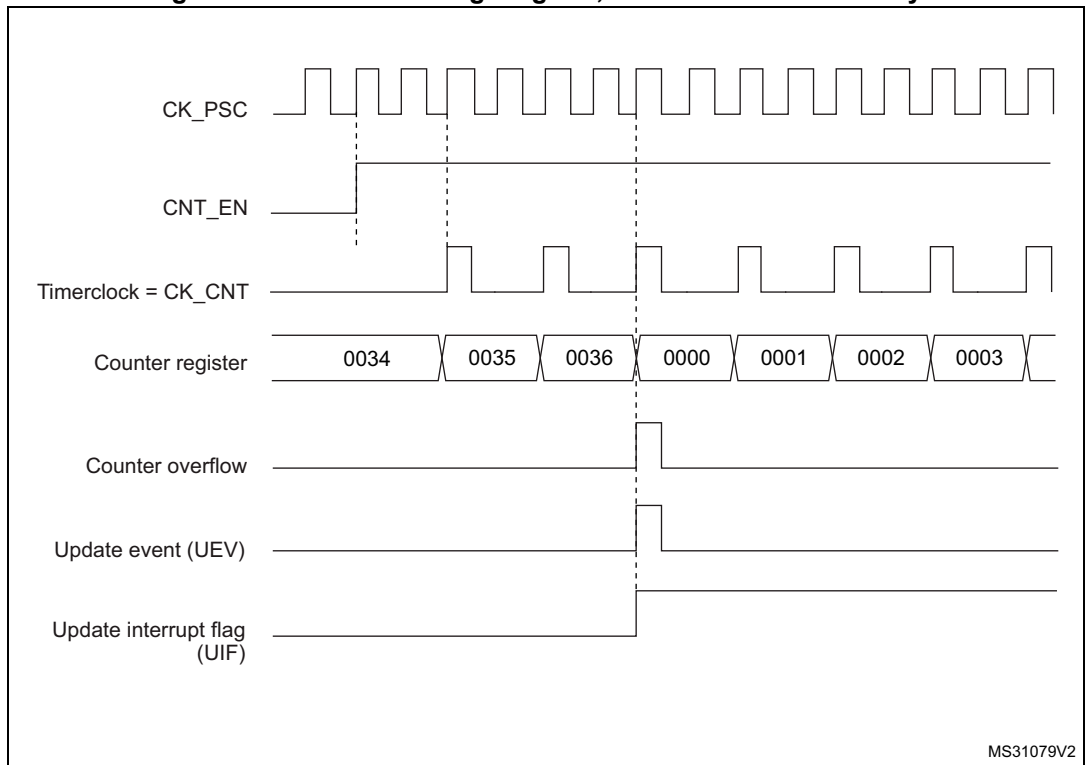


Figure 371. Counter timing diagram, internal clock divided by 4

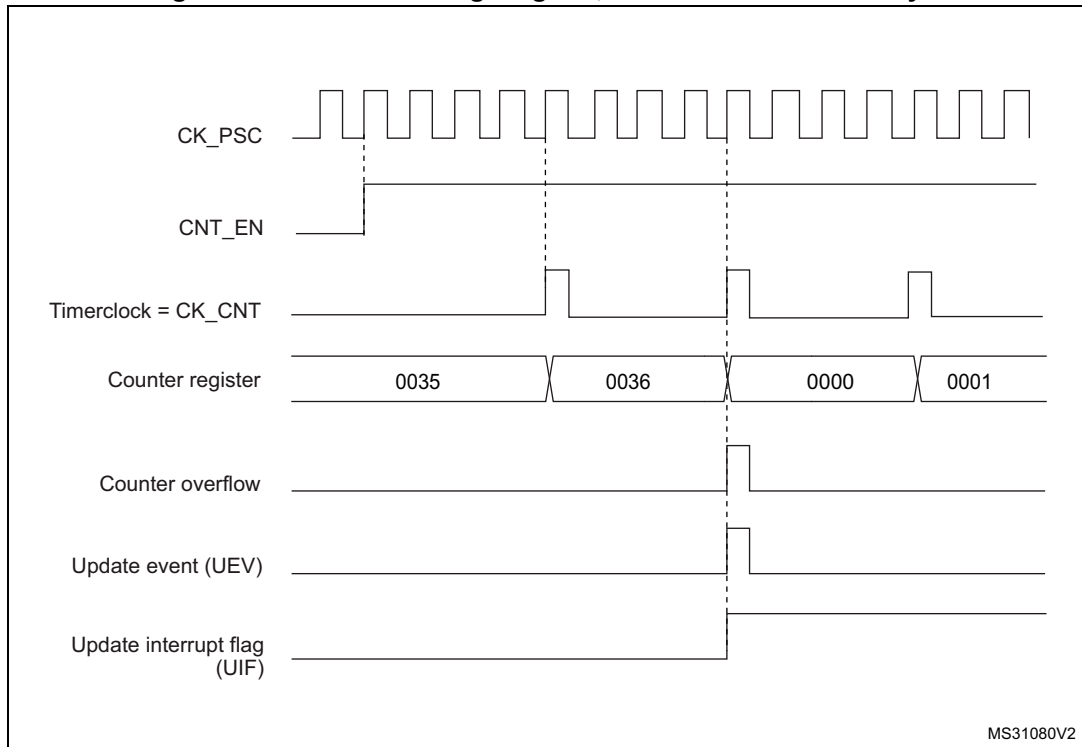
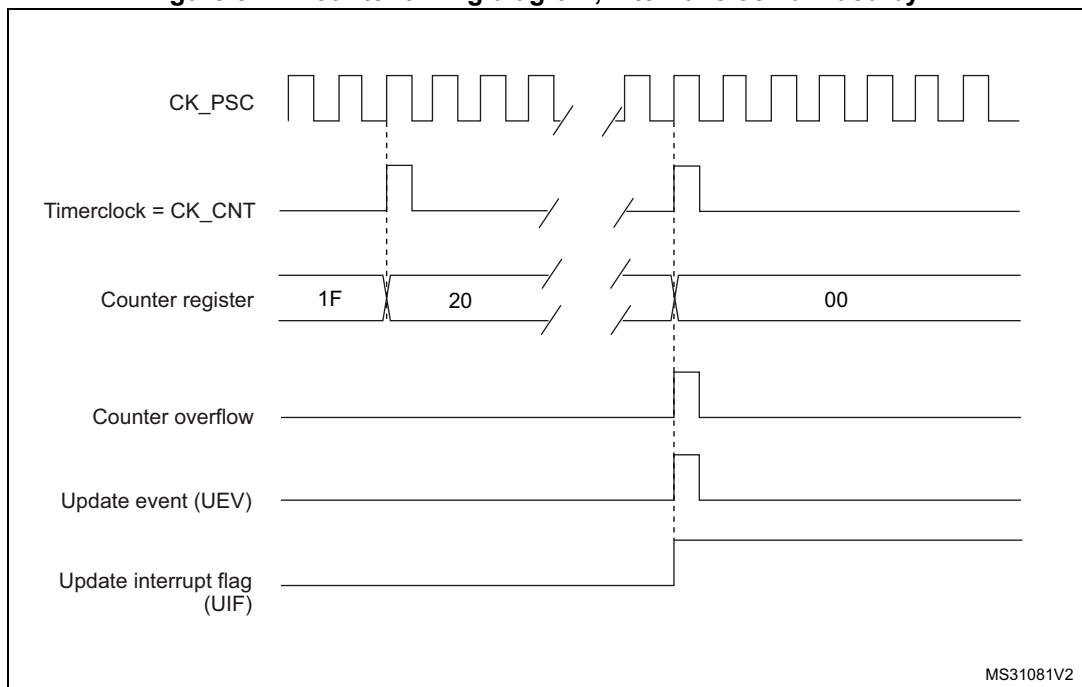
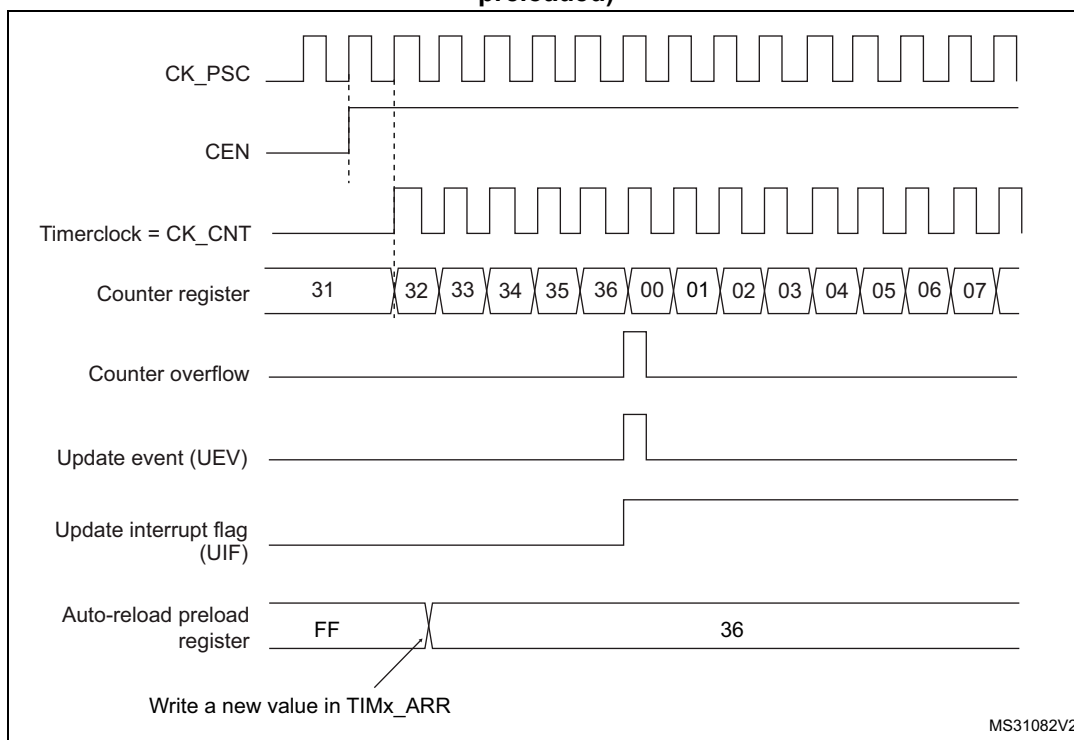


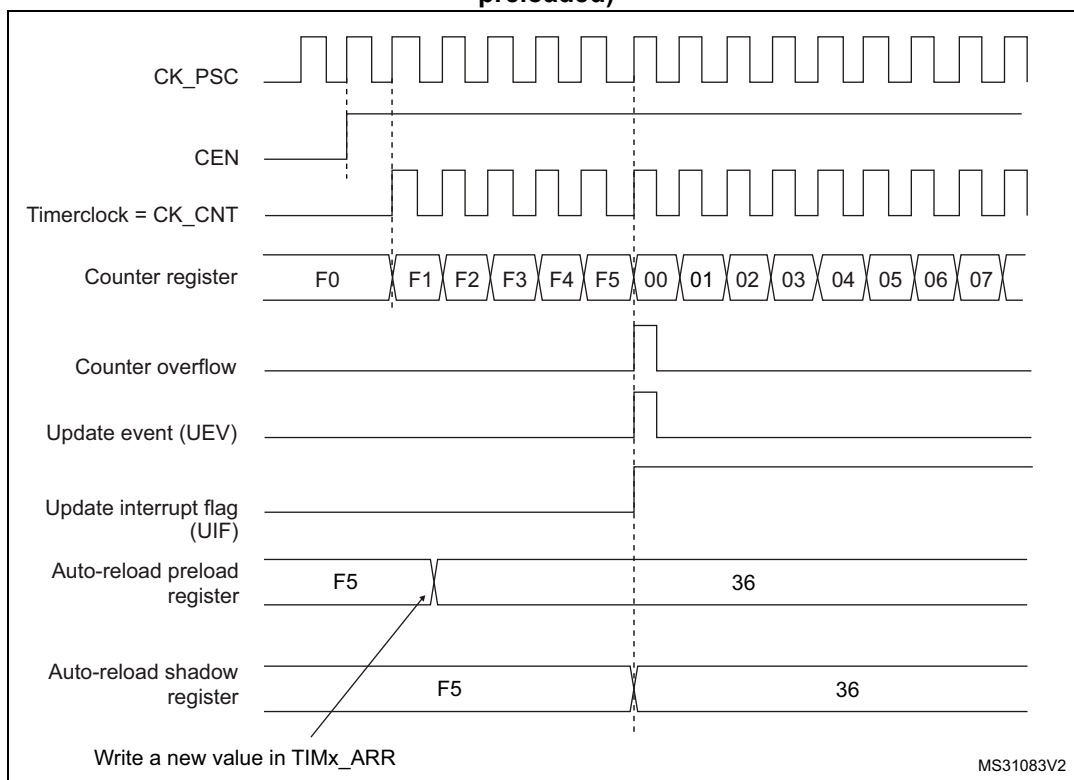
Figure 372. Counter timing diagram, internal clock divided by N



**Figure 373. Counter timing diagram, update event when ARPE=0 (TIMx\_ARR not preloaded)**



**Figure 374. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)**



### 36.4.3 Repetition counter

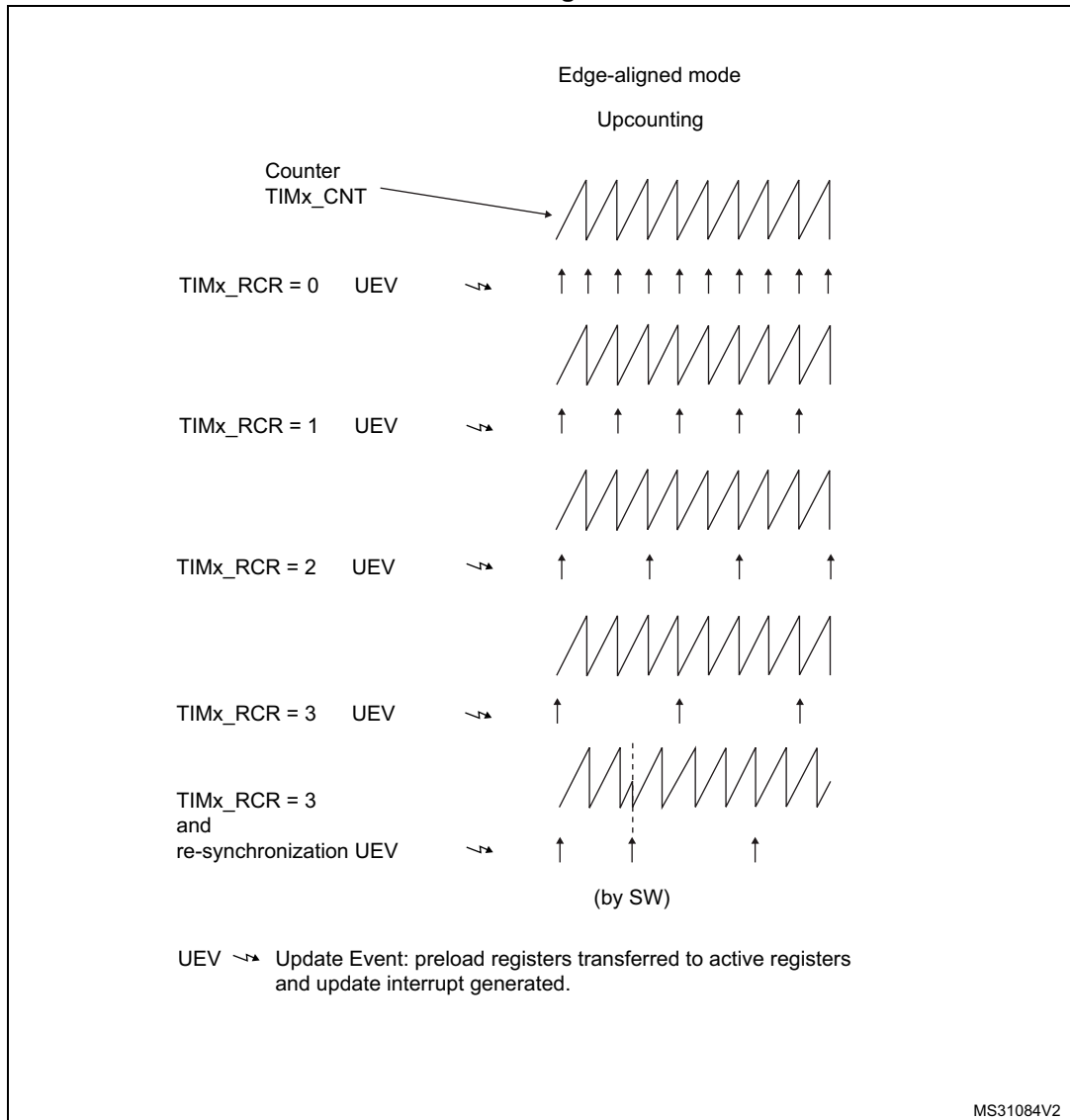
*Section 36.4.1: Time-base unit* describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx\_ARR auto-reload register, TIMx\_PSC prescaler register, but also TIMx\_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx\_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx\_RCR register value (refer to *Figure 375*). 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 375. Update rate examples depending on mode and TIMx\_RCR register settings



### 36.4.4 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK\_INT)
- External clock mode1: external input pin
- Internal trigger inputs (ITRx) (only for TIM15): using one timer as the prescaler for another timer, for example, you can configure TIM1 to act as a prescaler for TIM15. Refer to [Using one timer as prescaler for another timer on page 1229](#) for more details.

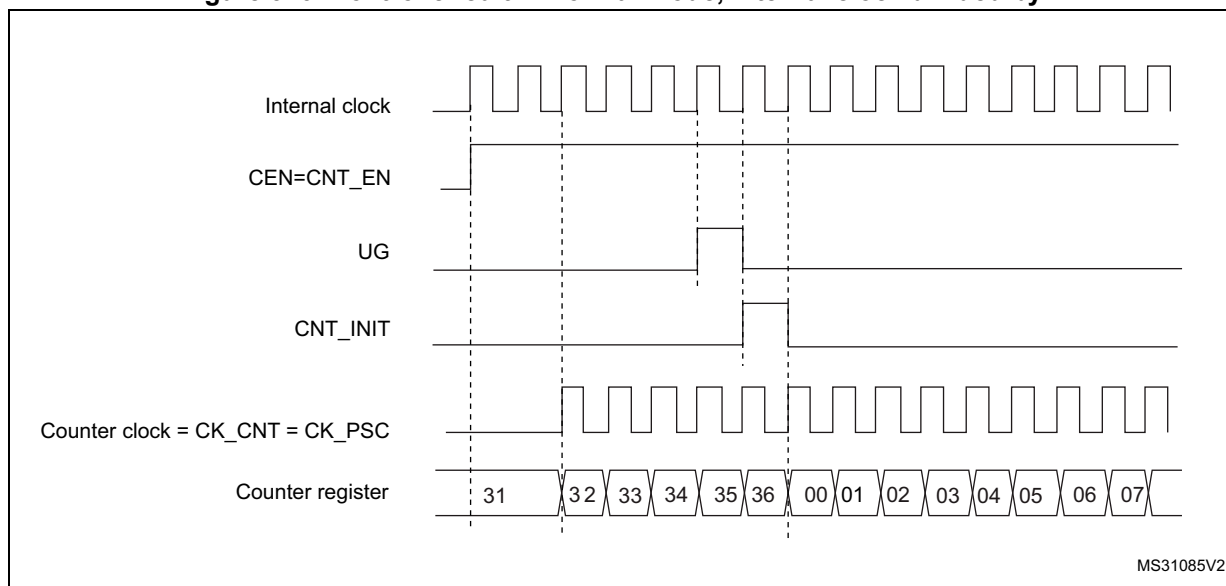
#### Internal clock source (CK\_INT)

If the slave mode controller is disabled (SMS=000), then the CEN (in the TIMx\_CR1 register) and UG bits (in the TIMx\_EGR register) are actual control bits and can be changed

only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK\_INT.

Figure 376 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

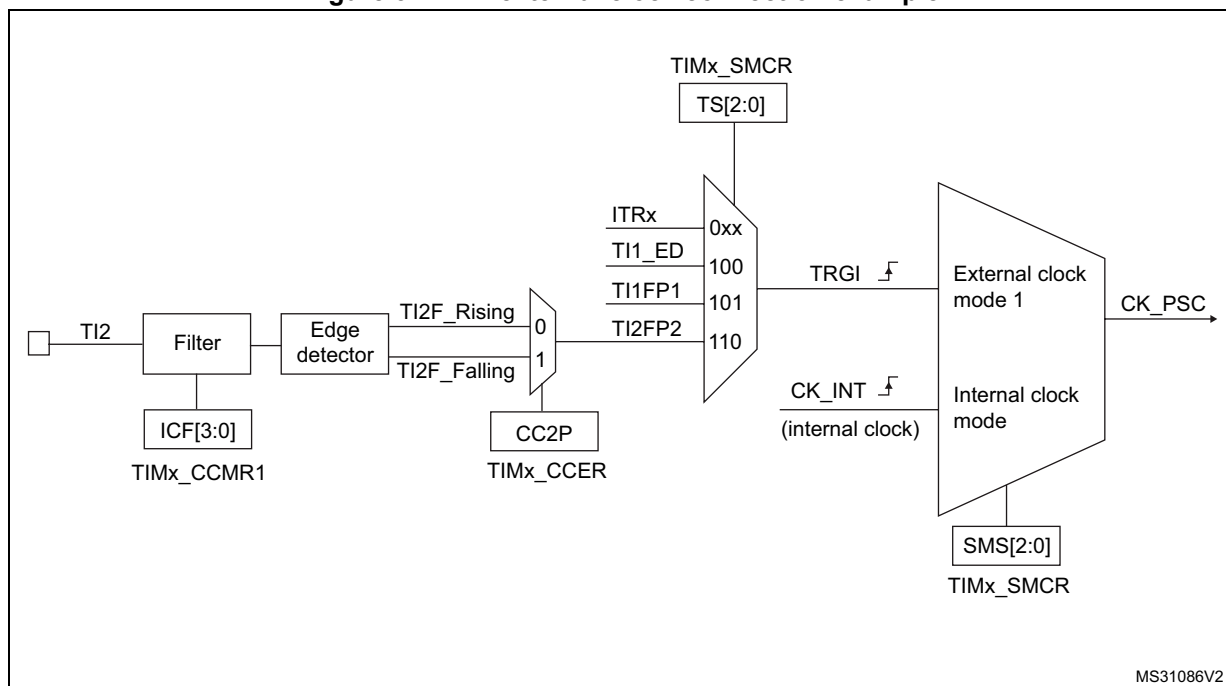
Figure 376. Control circuit in normal mode, internal clock divided by 1



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 377. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

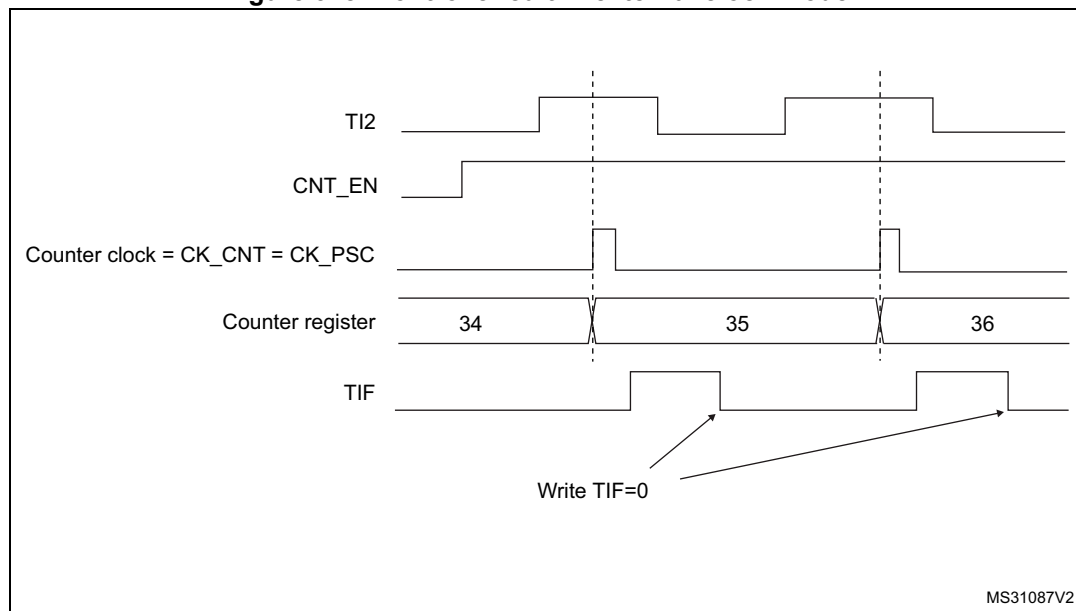
1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx\_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx\_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 in the TIMx\_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx\_SMCR register.
5. Select TI2 as the trigger input source by writing TS=110 in the TIMx\_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx\_CR1 register.

*Note:* The capture prescaler is not used for triggering, so you don't need to configure it.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

**Figure 378. Control circuit in external clock mode 1**



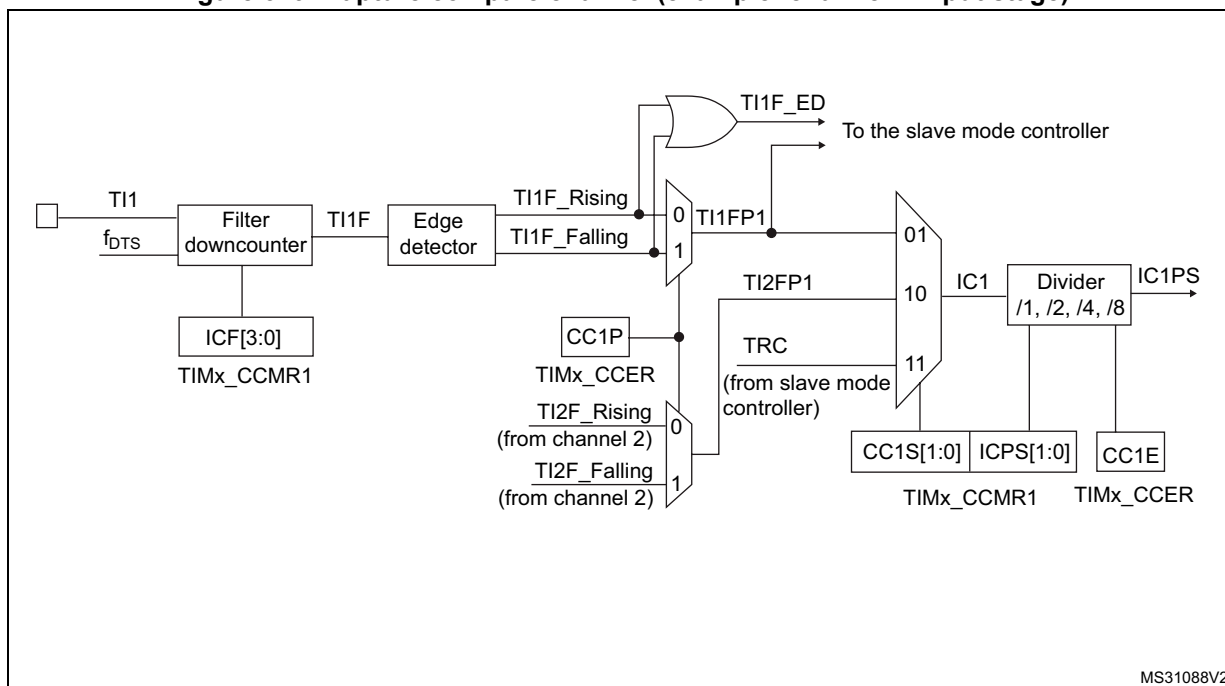
### 36.4.5 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

*Figure 379* to *Figure 382* 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 379. 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 380. Capture/compare channel 1 main circuit

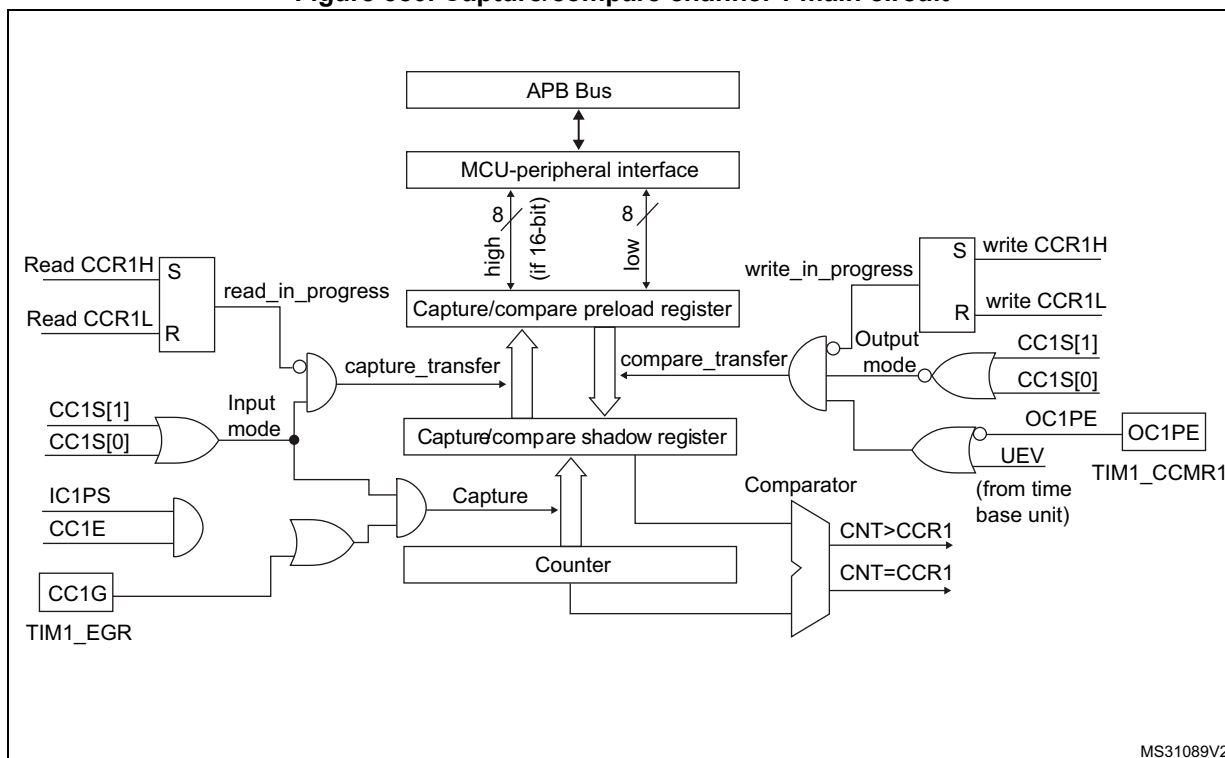
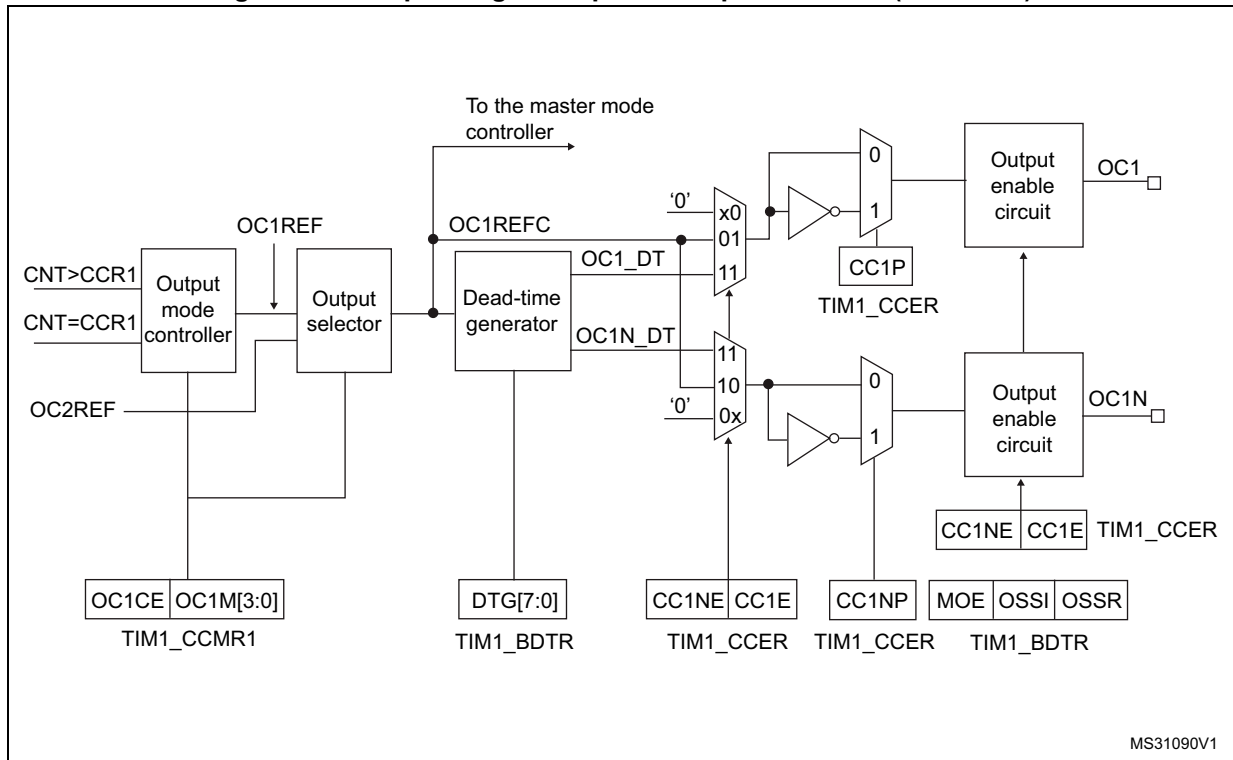


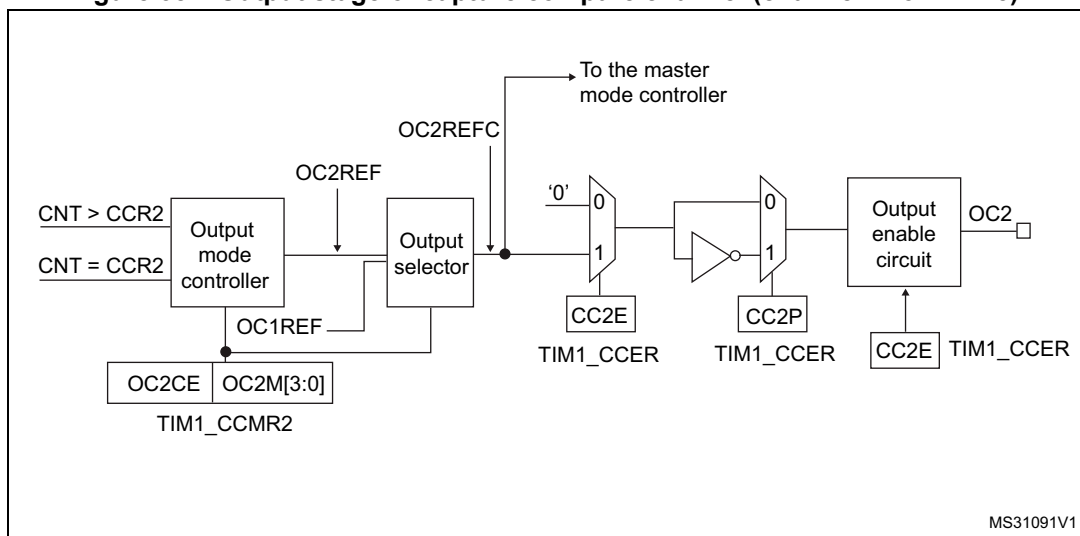


Figure 381. Output stage of capture/compare channel (channel 1)



MS31090V1

Figure 382. Output stage of capture/compare channel (channel 2 for TIM15)



MS31091V1

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

### 36.4.6 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx\_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx\_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx\_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx\_CCRx register. CCxOF is cleared when you write it to '0'.

The following example shows how to capture the counter value in TIMx\_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the active input: TIMx\_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx\_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx\_CCR1 register becomes read-only.
2. Program the input filter duration you need with respect to the signal you connect to the timer (when the input is one of the TIx (ICxF bits in the TIMx\_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at 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.
3. Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx\_CCER register (rising edge in this case).
4. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx\_CCMR1 register).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx\_CCER register.
6. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx\_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx\_DIER register.

When an input capture occurs:

- The TIMx\_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

*Note:* IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx\_EGR register.

### 36.4.7 PWM input mode (only for TIM15)

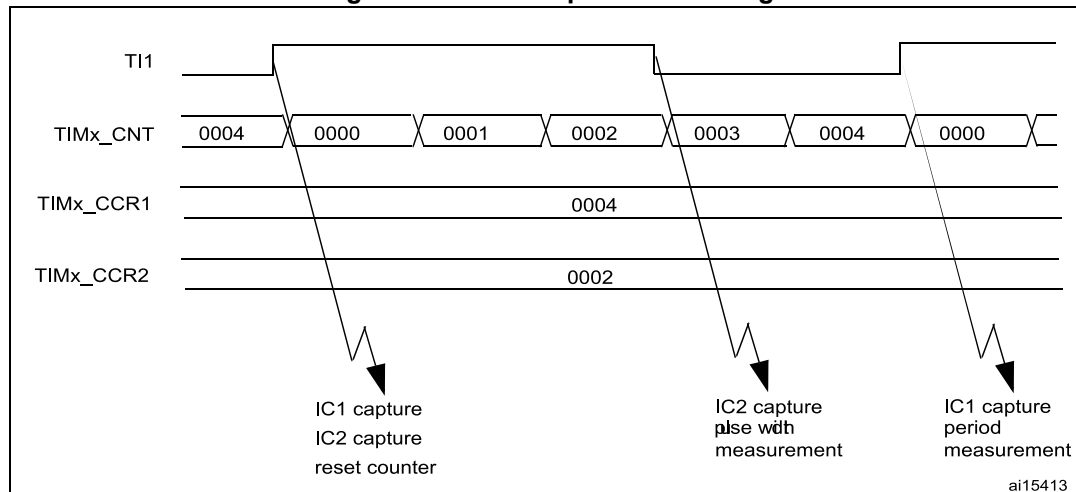
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, you can measure the period (in TIMx\_CCR1 register) and the duty cycle (in TIMx\_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK\_INT frequency and prescaler value):

1. Select the active input for TIMx\_CCR1: write the CC1S bits to 01 in the TIMx\_CCMR1 register (TI1 selected).
2. Select the active polarity for TI1FP1 (used both for capture in TIMx\_CCR1 and counter clear): write the CC1P and CC1NP bits to '0' (active on rising edge).
3. Select the active input for TIMx\_CCR2: write the CC2S bits to 10 in the TIMx\_CCMR1 register (TI1 selected).
4. Select the active polarity for TI1FP2 (used for capture in TIMx\_CCR2): write the CC2P and CC2NP bits to '1' (active on falling edge).
5. Select the valid trigger input: write the TS bits to 101 in the TIMx\_SMCR register (TI1FP1 selected).
6. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx\_SMCR register.
7. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx\_CCER register.

**Figure 383. 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.

### 36.4.8 Forced output mode

In output mode (CCxS bits = 00 in the TIMx\_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, you just need to write 101 in the OCxM bits in the corresponding TIMx\_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx\_CCMRx register.

Anyway, the comparison between the TIMx\_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

### 36.4.9 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx\_CCMRx register) and the output polarity (CCxP bit in the TIMx\_CCER register). The output pin can keep its level (OCxM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx\_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx\_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx\_DIER register, CCDS bit in the TIMx\_CR2 register for the DMA request selection).

The TIMx\_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx\_CCMRx register.

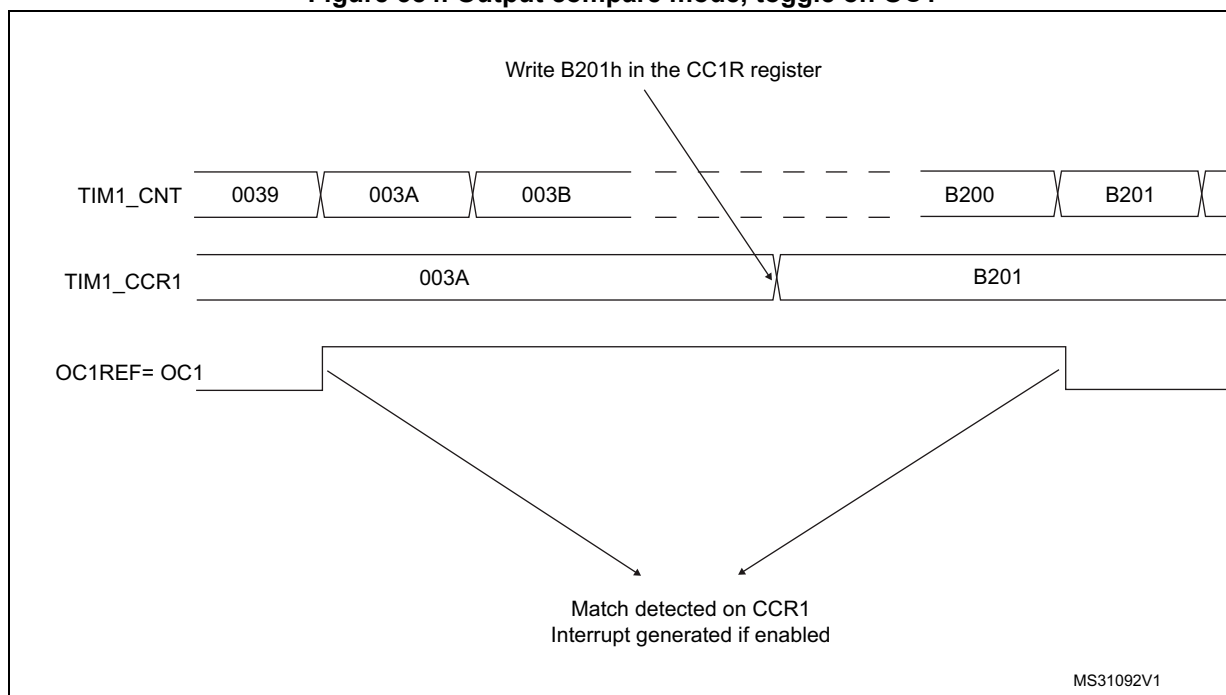
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

**Procedure**

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx\_ARR and TIMx\_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
  - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
  - Write OCxPE = 0 to disable preload register
  - Write CCxP = 0 to select active high polarity
  - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx\_CR1 register.

The TIMx\_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx\_CCRx shadow register is updated only at the next update event UEV). An example is given in [Figure 384](#).

**Figure 384. Output compare mode, toggle on OC1**



**36.4.10 PWM mode**

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx\_ARR register and a duty cycle determined by the value of the TIMx\_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx\_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx\_EGR register.

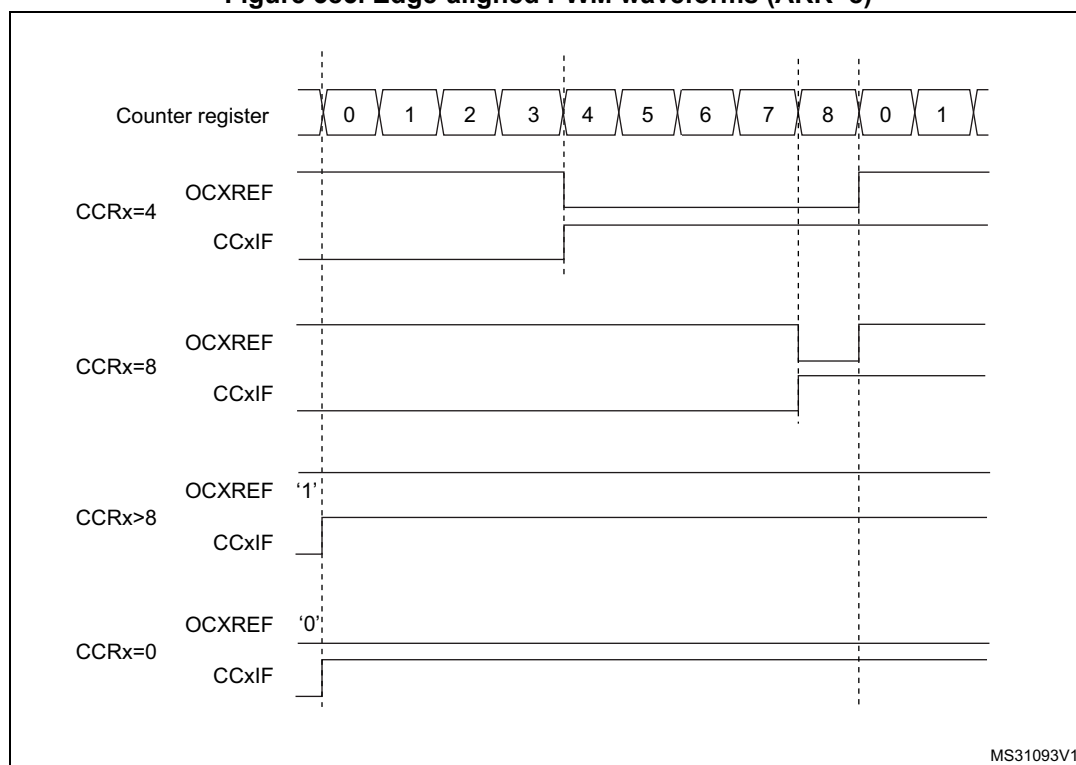
OCx polarity is software programmable using the CCxP bit in the TIMx\_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx\_CCER and TIMx\_BDTR registers). Refer to the TIMx\_CCER register description for more details.

In PWM mode (1 or 2), TIMx\_CNT and TIMx\_CCRx are always compared to determine whether  $TIMx\_CCRx \leq TIMx\_CNT$  or  $TIMx\_CNT \leq TIMx\_CCRx$  (depending on the direction of the counter).

The TIM15/TIM16/TIM17 are capable of upcounting only. Refer to [Upcounting mode on page 1266](#).

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 385](#) shows some edge-aligned PWM waveforms in an example where  $TIMx\_ARR=8$ .

**Figure 385. Edge-aligned PWM waveforms (ARR=8)**



### 36.4.11 Combined PWM mode (TIM15 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx\_ARR register, the duty cycle and delay are determined

by the two TIMx\_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by the TIMx\_CCR1 and TIMx\_CCR2 registers

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing '1100' (Combined PWM mode 1) or '1101' (Combined PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register.

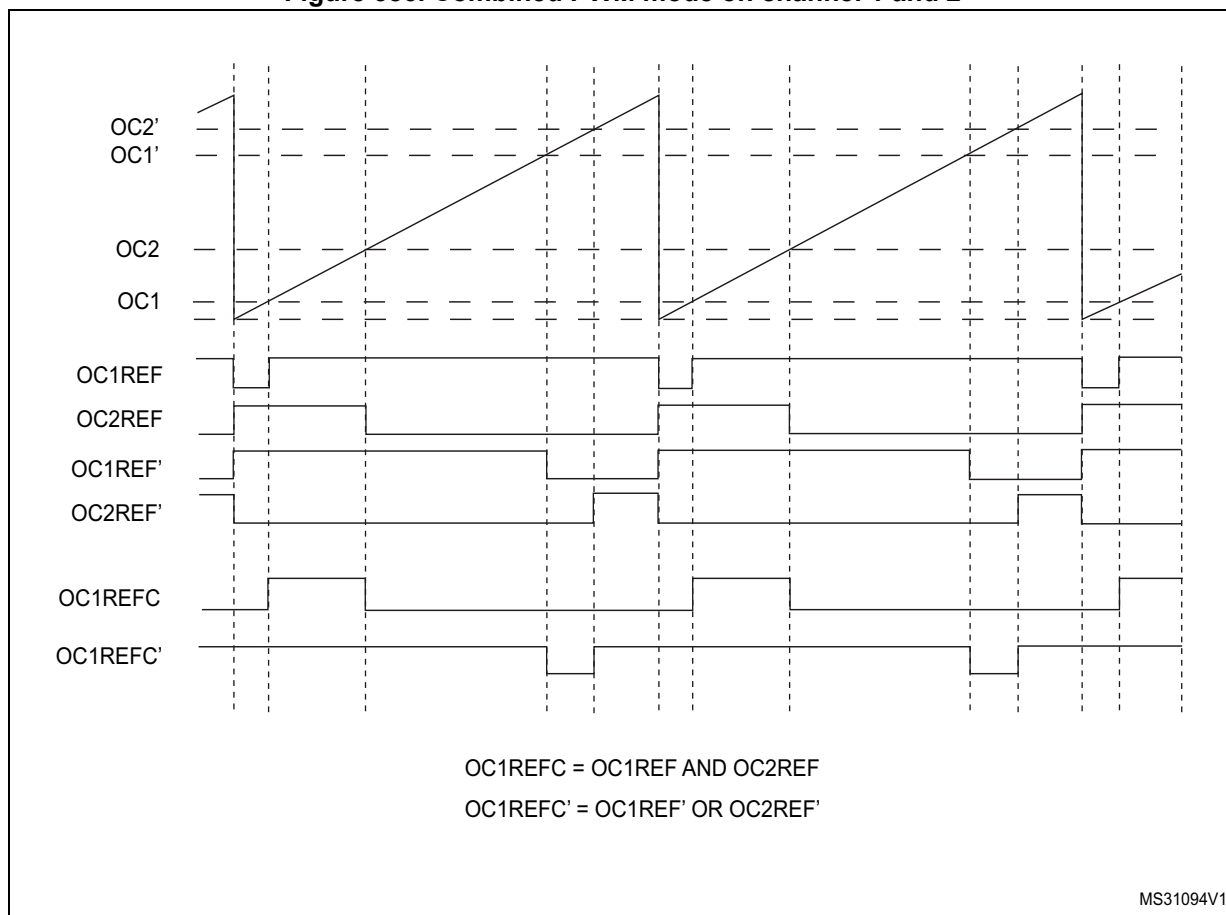
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

*Note:* The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 386 represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,

**Figure 386. Combined PWM mode on channel 1 and 2**



### 36.4.12 Complementary outputs and dead-time insertion

The TIM15/TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and you have to adjust it depending on the devices you have connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

You can select the polarity of the outputs (main output OCx or complementary OCxN) independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx\_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx\_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx\_BDTR and TIMx\_CR2 registers. Refer to [Table 229: Output control bits for complementary OCx and OCxN channels with break feature \(TIM16/17\) on page 1334](#) for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

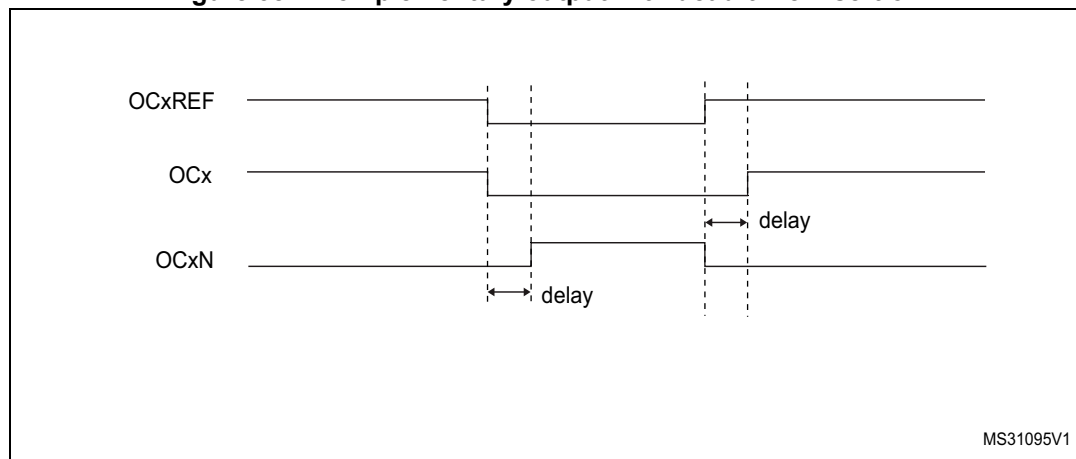
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- 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.

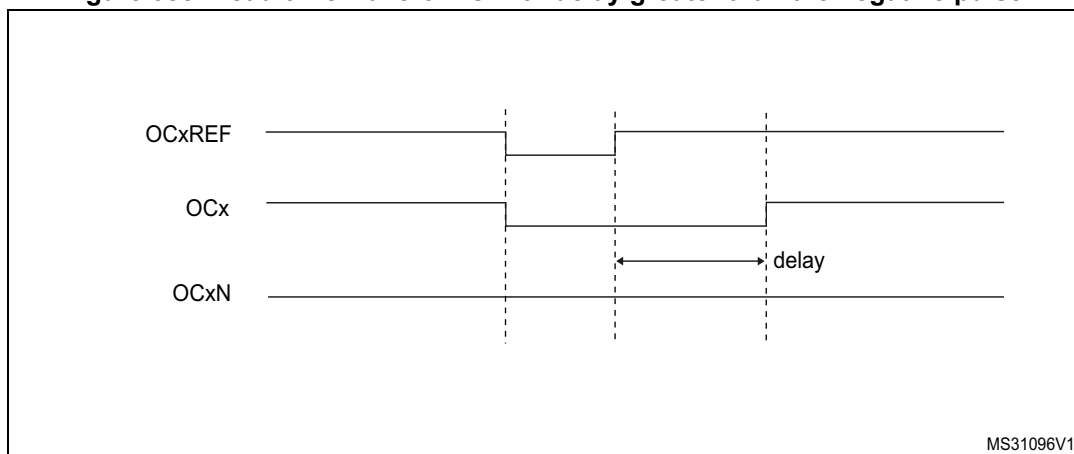
The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

**Figure 387. Complementary output with dead-time insertion.**



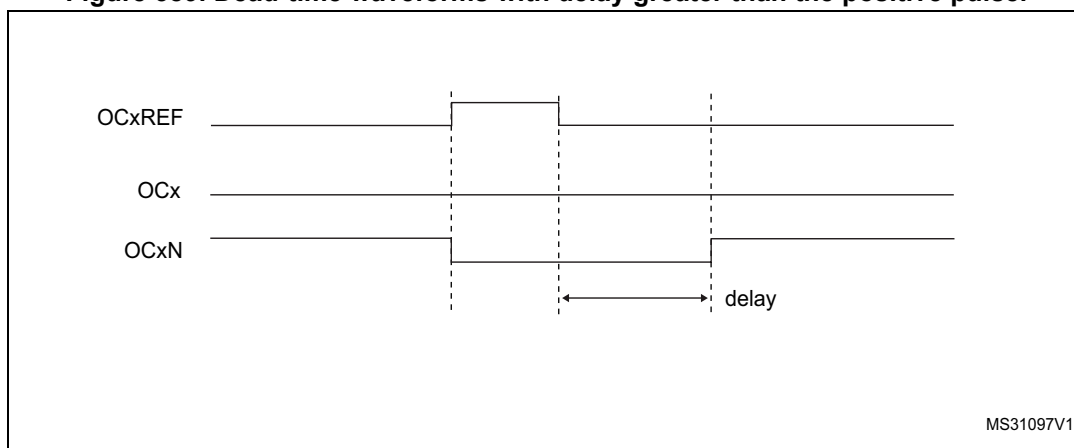


**Figure 388. Dead-time waveforms with delay greater than the negative pulse.**



MS31096V1

**Figure 389. Dead-time waveforms with delay greater than the positive pulse.**



MS31097V1

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 36.6.13: TIM16/TIM17 break and dead-time register \(TIMx\\_BDTR\) on page 1337](#) for delay calculation.

### Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx\_CCER register.

This allows you to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

*Note:* When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

### 36.4.13 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM15/TIM16/TIM17 timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx\_BDTR register allows to enable /disable the outputs by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx\_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx\_CR2 register which are setting the output shut-down level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to [Table 227: Output control bits for complementary OCx and OCxN channels with break feature \(TIM15\) on page 1313](#) for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx\_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx\_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if you write MOE to 1 whereas it was low, you must insert a delay (dummy instruction) before reading it correctly. This is because you write the asynchronous signal and read the synchronous signal.

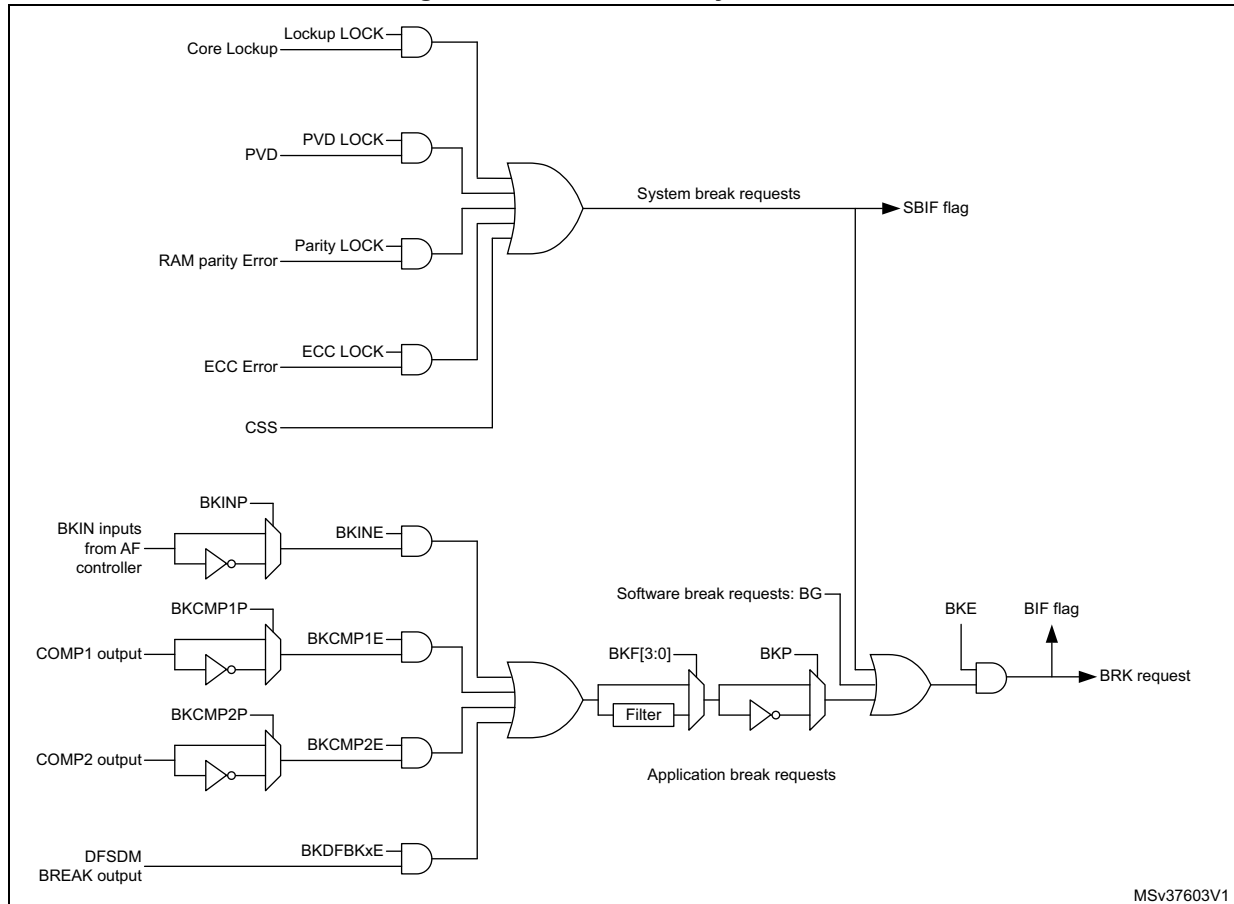
The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx\_OR2 register.

The sources for break (BRK) channel are:

- An external source connected to one of the BKIN pin (as per selection done in the AFIO controller), with polarity selection and optional digital filtering
- An internal source:
  - the Cortex®-M4 LOCKUP output
  - the PVD output
  - the SRAM parity error signal
  - a flash ECC error
  - a clock failure event generated by the CSS detector
  - the output from a comparator, with polarity selection and optional digital filtering
  - the analog watchdog output of the DFSDM1 peripheral

Break events can also be generated by software using BG bit in the TIMx\_EGR register. All sources are ORed before entering the timer BRK inputs, as per [Figure 390](#) below.

**Figure 390. Break 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 a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the AFIO controller (selected by the OSS1 bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx\_CR2 register as soon as MOE=0. If OSS1=0, the timer releases the output control (taken over by the AFIO controller) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their

- active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2  $ck\_tim$  clock cycles).
- If OSSI=0 then the timer releases the enable outputs (taken over by the AFIO controller which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
  - The break status flag (BIF bit in the TIMx\_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx\_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx\_DIER register is set.
  - If the AOE bit in the TIMx\_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until you write it to '1' again. In this case, it can be used for security and you can connect the break input to an alarm from power drivers, thermal sensors or any security components.

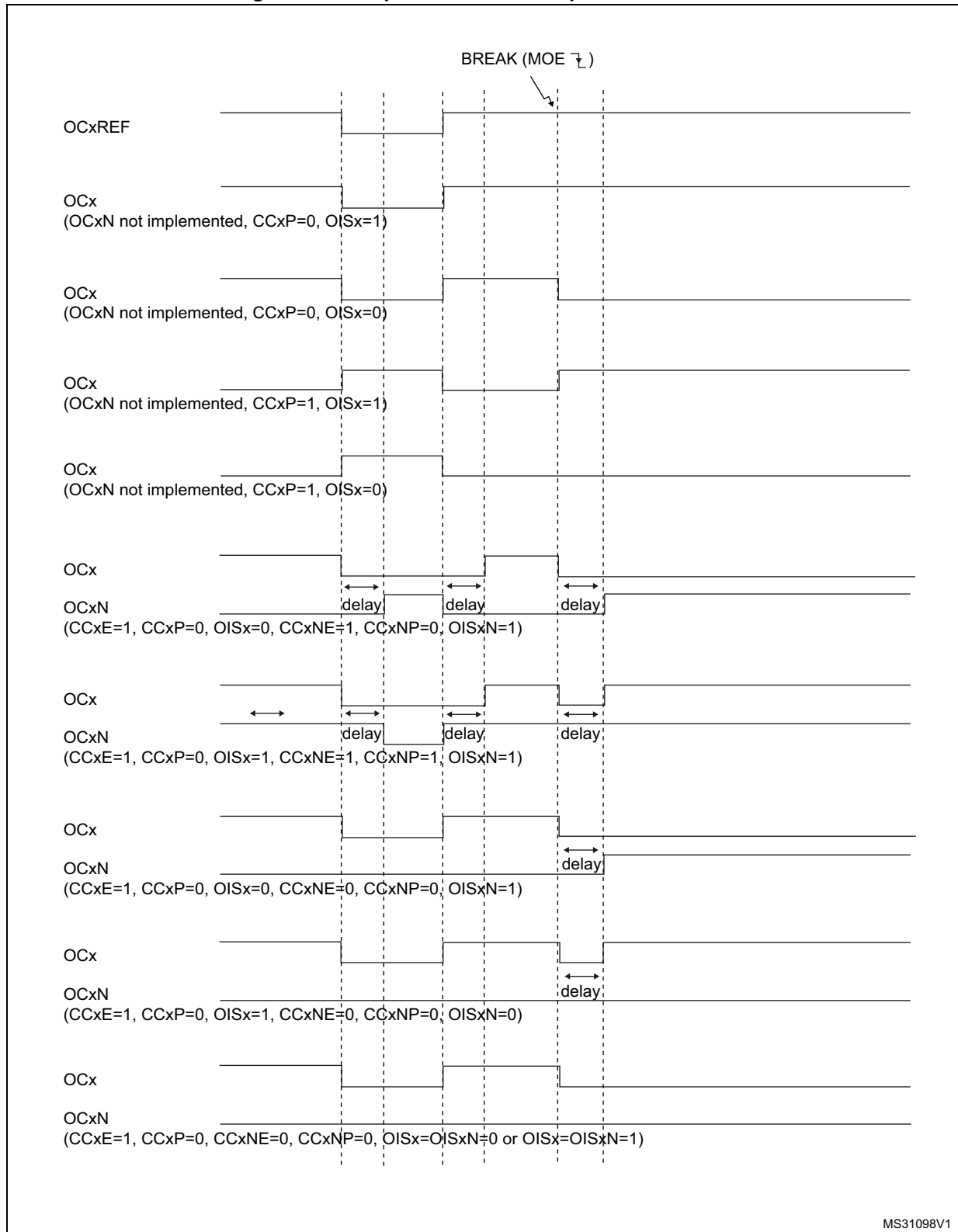
*Note:* The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx\_BDTR Register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows you to freeze the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). You can choose from 3 levels of protection selected by the LOCK bits in the TIMx\_BDTR register. Refer to [Section 36.6.13: TIM16/TIM17 break and dead-time register \(TIMx\\_BDTR\) on page 1337](#). The LOCK bits can be written only once after an MCU reset.

The [Figure 391](#) shows an example of behavior of the outputs in response to a break.

Figure 391. Output behavior in response to a break



### 36.4.14 Bidirectional break inputs

The TIM15/TIM16/TIM17 are featuring bidirectional break I/Os, as represented on [Figure 392](#).

They allow to have:

- 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 will be active even if the BKDSRM bit is set and the open drain control is released. This will prevent 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 225](#))

**Table 225. 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

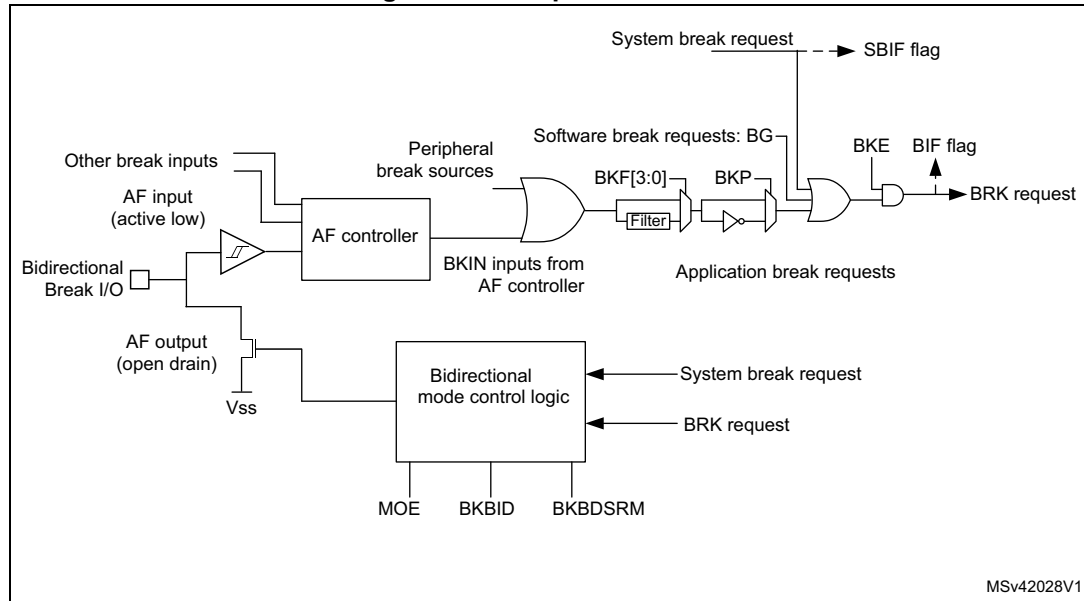
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

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 392. Output redirection**



### 36.4.15 One-pulse mode

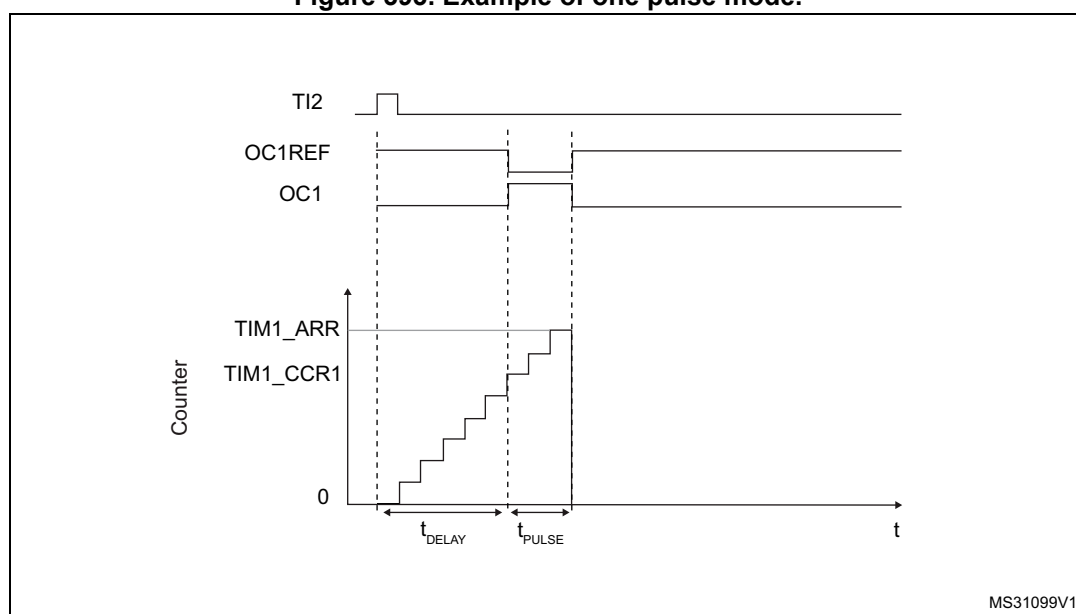
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx\_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- $CNT < CCRx \leq ARR$  (in particular,  $0 < CCRx$ )

Figure 393. Example of one pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of  $t_{PULSE}$  and after a delay of  $t_{DELAY}$  as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Map TI2FP2 to TI2 by writing  $CC2S='01'$  in the TIMx\_CCMR1 register.
2. TI2FP2 must detect a rising edge, write  $CC2P='0'$  and  $CC2NP='0'$  in the TIMx\_CCER register.
3. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing  $TS='110'$  in the TIMx\_SMCR register.
4. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx\_SMCR register (trigger mode).



The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The  $t_{\text{DELAY}}$  is defined by the value written in the TIMx\_CCR1 register.
- The  $t_{\text{PULSE}}$  is defined by the difference between the auto-reload value and the compare value (TIMx\_ARR - TIMx\_CCR1).
- Let's say you want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M=111 in the TIMx\_CCMR1 register. You can optionally enable the preload registers by writing OC1PE='1' in the TIMx\_CCMR1 register and ARPE in the TIMx\_CR1 register. In this case you have to write the compare value in the TIMx\_CCR1 register, the auto-reload value in the TIMx\_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

You only want 1 pulse, so you write '1' in the OPM bit in the TIMx\_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

Particular case: OCx fast enable

In One-pulse mode, the edge detection on Tlx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay  $t_{\text{DELAY}}$  min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx\_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 36.4.16 Retriggerable one pulse mode (OPM) (TIM15 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in [Section 36.4.15](#):

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

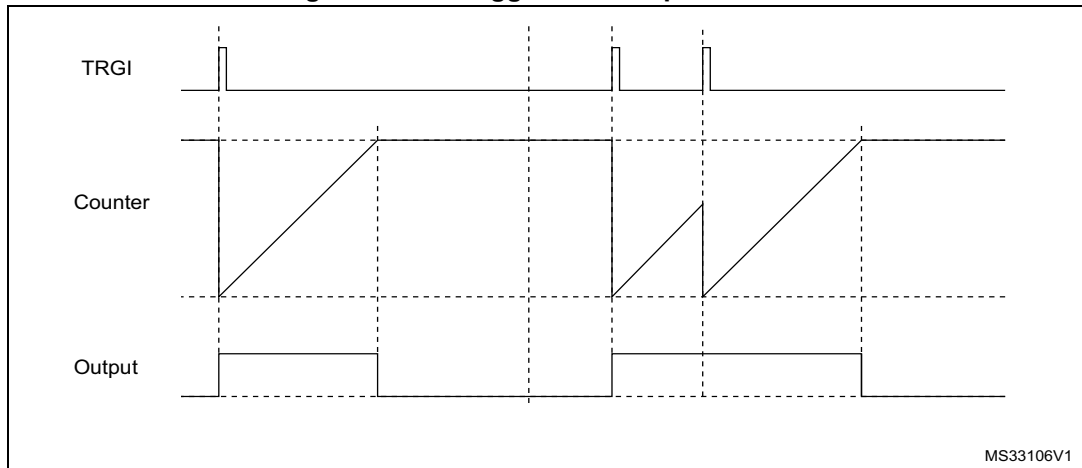
The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx\_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

*Note:* The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

*This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx\_CR1.*

Figure 394. Retriggerable one pulse mode



### 36.4.17 UIF bit remapping

The IUFREMAP bit in the TIMx\_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

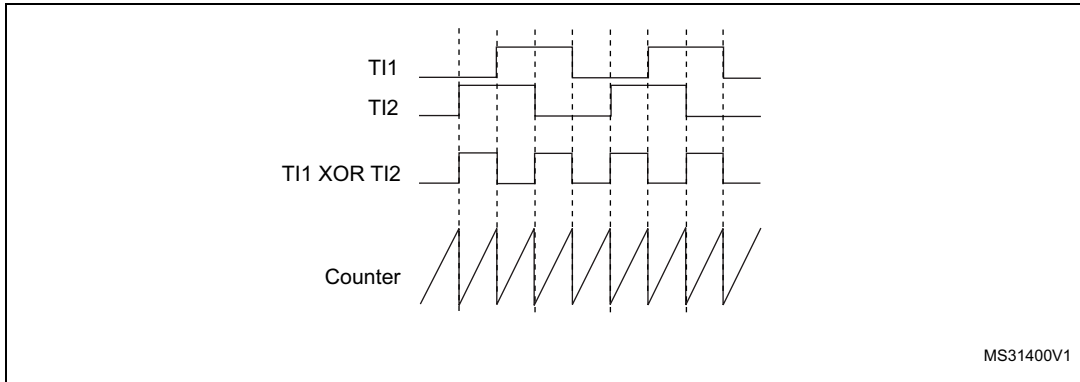
There is no latency between the assertions of the UIF and UIFCPY flags.

**36.4.18 Timer input XOR function (TIM15 only)**

The TI1S bit in the TIMx\_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins TIMx\_CH1 and TIMx\_CH2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in [Figure 395](#).

**Figure 395. Measuring time interval between edges on 2 signals**



### 36.4.19 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

#### Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx\_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx\_ARR, TIMx\_CCRx) are updated.

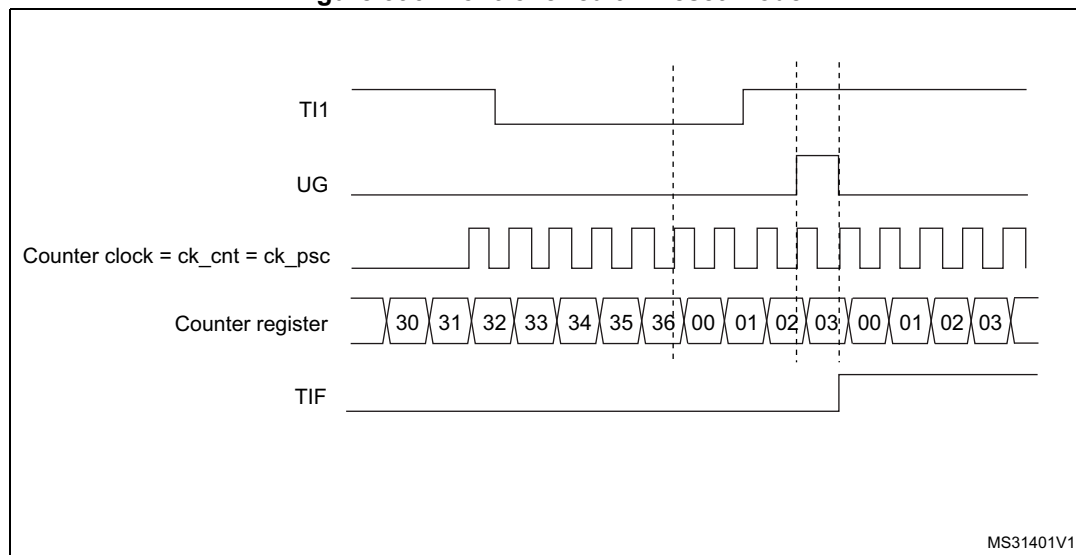
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx\_CCMR1 register. Write CC1P='0' and CC1NP='0' in the TIMx\_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx\_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx\_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx\_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx\_DIER register).

The following figure shows this behavior when the auto-reload register TIMx\_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 396. Control circuit in reset mode



MS31401V1

**Slave mode: Gated mode**

The counter can be enabled depending on the level of a selected input.

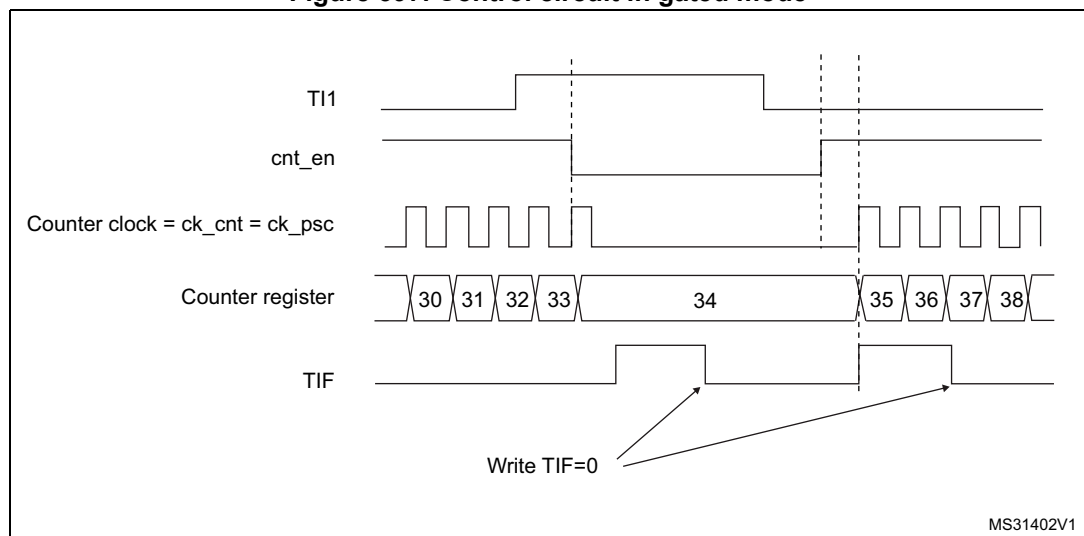
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S=01 in TIMx\_CCMR1 register. Write CC1P=1 and CC1NP = '0' in the TIMx\_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx\_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx\_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx\_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

**Figure 397. Control circuit in gated mode**



### Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

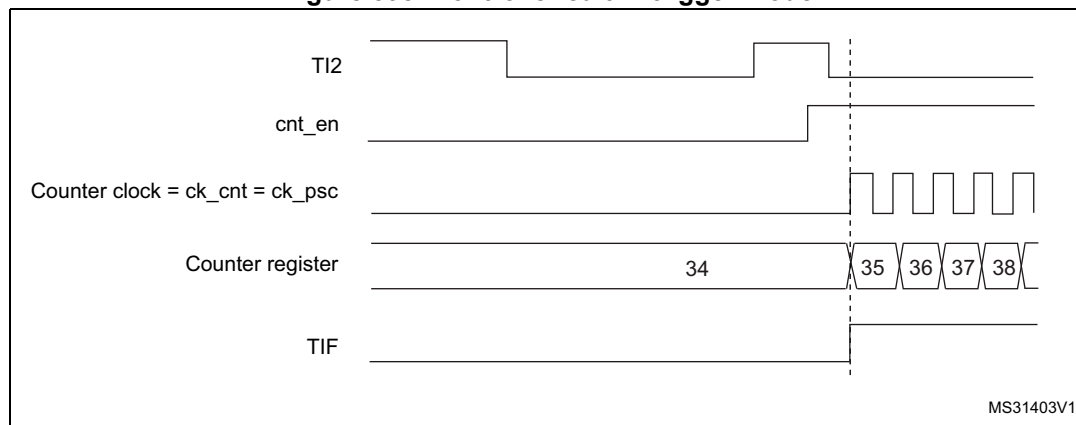
In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx\_CCMR1 register. Write CC2P='1' and CC2NP='0' in the TIMx\_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS=110 in the TIMx\_SMCR register. Select TI2 as the input source by writing TS=110 in the TIMx\_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

**Figure 398. Control circuit in trigger mode**



### 36.4.20 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.

### 36.4.21 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.

### 36.4.22 Timer synchronization (TIM15)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to [Section 35.3.19: Timer synchronization](#) for details.

*Note:* The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

### 36.4.23 Debug mode

When the microcontroller enters debug mode (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 51.16.2: Debug support for timers, RTC, watchdog, bxCAN and I<sup>2</sup>C](#).

When the microcontroller enters debug mode (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 51.16.2: Debug support for timers, RTC, watchdog, bxCAN and I<sup>2</sup>C](#).

For safety purposes, when the counter is stopped (DBG\_TIMx\_STOP = 1), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.



## 36.5 TIM15 registers

Refer to [Section 1.2](#) for a list of abbreviations used in register descriptions.

### 36.5.1 TIM15 control register 1 (TIM15\_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

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

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx\_CNT register bit 31.

1: Remapping enabled. UIF status bit is copied to TIMx\_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (CK\_INT) frequency and the dead-time and sampling clock ( $t_{DTS}$ ) used by the dead-time generators and the digital filters (Tix)

00:  $t_{DTS} = t_{CK\_INT}$

01:  $t_{DTS} = 2 * t_{CK\_INT}$

10:  $t_{DTS} = 4 * t_{CK\_INT}$

11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx\_ARR register is not buffered

1: TIMx\_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

*Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*

### 36.5.2 TIM15 control register 2 (TIM15\_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

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

Bit 10 **OIS2**: Output idle state 2 (OC2 output)

0: OC2=0 when MOE=0

1: OC2=1 when MOE=0

*Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIMx\_BKR register).*

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0

1: OC1N=1 after a dead-time when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BKR register).*

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BKR register).*

Bit 7 **TI1S**: TI1 selection

- 0: The TIMx\_CH1 pin is connected to TI1 input
- 1: The TIMx\_CH1, CH2 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx\_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter Enable signal CNT\_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx\_SMCR register).

010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).

100: **Compare** - OC1REF signal is used as trigger output (TRGO).

101: **Compare** - OC2REF signal is used as trigger output (TRGO).

Bit 3 **CCDS**: Capture/compare DMA selection

- 0: CCx DMA request sent when CCx event occurs
- 1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

*Note: This bit acts only on channels that have a complementary output.*

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).

*Note: This bit acts only on channels that have a complementary output.*

### 36.5.3 TIM15 slave mode control register (TIM15\_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

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

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

Bit 16 **SMS[3]**: Slave mode selection - bit 3  
 Refer to SMS description - bits 2:0.

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

Bit 7 **MSM**: Master/slave mode  
 0: No action  
 1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]**: Trigger selection  
 This bit field selects the trigger input to be used to synchronize the counter.  
 000: Internal Trigger 0 (ITR0)  
 001: Internal Trigger 1 (ITR1)  
 010: Internal Trigger 2 (ITR2)  
 011: Internal Trigger 3 (ITR3)  
 100: TI1 Edge Detector (TI1F\_ED)  
 101: Filtered Timer Input 1 (TI1FP1)  
 110: Filtered Timer Input 2 (TI2FP2)  
 See [Table 226: TIMx Internal trigger connection on page 1303](#) for more details on ITRx meaning for each Timer.

*Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.*

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description).

0000: Slave mode disabled - if CEN = '1' then the prescaler is clocked directly by the internal clock.

0001: Reserved

0010: Reserved

0011: Reserved

0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Other codes: reserved.

*Note: The gated mode must not be used if TI1F\_ED is selected as the trigger input (TS='100'). Indeed, TI1F\_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.*

*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 226. TIMx Internal trigger connection**

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM15	TIM1	TIM3	TIM16 OC1	TIM17 OC1

### 36.5.4 TIM15 DMA/interrupt enable register (TIM15\_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMD E	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	rw	rw			rw	rw	rw	rw	rw	rw			rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

0: Trigger DMA request disabled

1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable

0: COM DMA request disabled

1: COM DMA request enabled

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable  
 0: CC2 DMA request disabled  
 1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable  
 0: CC1 DMA request disabled  
 1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable  
 0: Update DMA request disabled  
 1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable  
 0: Break interrupt disabled  
 1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable  
 0: Trigger interrupt disabled  
 1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable  
 0: COM interrupt disabled  
 1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable  
 0: CC2 interrupt disabled  
 1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable  
 0: CC1 interrupt disabled  
 1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable  
 0: Update interrupt disabled  
 1: Update interrupt enabled

### 36.5.5 TIM15 status register (TIM15\_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0		rc_w0	rc_w0	rc_w0			rc_w0	rc_w0	rc_w0

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

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag  
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag  
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.  
0: No overcapture has been detected  
1: The counter value has been captured in TIMx\_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag  
This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.  
0: No break event occurred  
1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag  
This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.  
0: No trigger event occurred  
1: Trigger interrupt pending

Bit 5 **COMIF**: COM interrupt flag  
This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.  
0: No COM event occurred  
1: COM interrupt pending

Bits 4:3 Reserved, must be kept at reset value.

- Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag  
refer to CC1IF description
- Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag  
  - If channel CC1 is configured as output:** This flag is set by hardware when the counter matches the compare value. It is cleared by software.
  - 0: No match.
  - 1: The content of the counter TIMx\_CNT matches the content of the TIMx\_CCR1 register. When the contents of TIMx\_CCR1 are greater than the contents of TIMx\_ARR, the CC1IF bit goes high on the counter overflow.
  - If channel CC1 is configured as input:** This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx\_CCR1 register.
  - 0: No input capture occurred
  - 1: The counter value has been captured in TIMx\_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)
- Bit 0 **UIF**: Update interrupt flag  
  - This bit is set by hardware on an update event. It is cleared by software.
  - 0: No update occurred.
  - 1: Update interrupt pending. This bit is set by hardware when the registers are updated:
    - At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx\_CR1 register.
    - When CNT is reinitialized by software using the UG bit in TIMx\_EGR register, if URS=0 and UDIS=0 in the TIMx\_CR1 register.
    - When CNT is reinitialized by a trigger event (refer to [Section 36.5.3: TIM15 slave mode control register \(TIM15\\_SMCR\)](#)), if URS=0 and UDIS=0 in the TIMx\_CR1 register.

### 36.5.6 TIM15 event generation register (TIM15\_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
								w	w	rw			w	w	w

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

- Bit 7 **BG**: Break generation  
  - This bit is set by software in order to generate an event, it is automatically cleared by hardware.
  - 0: No action
  - 1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.
- Bit 6 **TG**: Trigger generation  
  - This bit is set by software in order to generate an event, it is automatically cleared by hardware.
  - 0: No action
  - 1: The TIF flag is set in TIMx\_SR register. Related interrupt or DMA transfer can occur if enabled



Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

*Note: This bit acts only on channels that have a complementary output.*

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/Compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

**If channel CC1 is configured as output:**

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

**If channel CC1 is configured as input:**

The current value of the counter is captured in TIMx\_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

### 36.5.7 TIM15 capture/compare mode register 1 (TIM15\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							Res.								Res.
							rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		Res.	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
IC2F[3:0]			IC2PSC[1:0]					IC1F[3:0]			IC1PSC[1:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

**Output compare mode:**

Bits 31:25 Reserved, always read as 0

Bit 24 **OC2M[3]**: Output Compare 2 mode - bit 3

Bits 23:17 Reserved, always read as 0

Bit 16 **OC1M[3]**: Output Compare 1 mode - bit 3  
refer to OC1M description on bits 6:4

Bit 15 Reserved, always read as 0

Bits 14:12 **OC2M[2:0]**: Output Compare 2 mode

Bit 11 **OC2PE**: Output Compare 2 preload enable

Bit 10 **OC2FE**: Output Compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, IC2 is mapped on TI2.

10: CC2 channel is configured as input, IC2 is mapped on TI1.

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx\_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx\_CCER).*

Bit 7 Reserved, always read as 0

Bits 6:4 **OC1M**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx\_CCR1 and the counter TIMx\_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0011: Toggle - OC1REF toggles when TIMx\_CNT=TIMx\_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx\_CNT<TIMx\_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx\_CNT<TIMx\_CCR1 else active.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved

1011: Reserved

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Reserved,

1111: Reserved,

*Note: 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S='00' (the channel is configured in output).*

*2: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.*

*3: On channels that have a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.*

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx\_CCR1 disabled. TIMx\_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx\_CCR1 enabled. Read/Write operations access the preload register. TIMx\_CCR1 preload value is loaded in the active register at each update event.

*Note: 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S='00' (the channel is configured in output).*

*2: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx\_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, IC1 is mapped on TI1.

10: CC1 channel is configured as input, IC1 is mapped on TI2.

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx\_CCER).*

**Input capture mode**

Bits 31:16 Reserved, always read as 0

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx\_CCER).*

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N 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**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx\_CCER register).

- 00: no prescaler, capture is done each time an edge is detected on the capture input
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, IC1 is mapped on TI1
- 10: CC1 channel is configured as input, IC1 is mapped on TI2
- 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx\_CCER).

### 36.5.8 TIM15 capture/compare enable register (TIM15\_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								rw		rw	rw	rw	rw	rw	rw

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

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity  
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

- Bit 5 **CC2P**: Capture/Compare 2 output polarity  
Refer to CC1P description
- Bit 4 **CC2E**: Capture/Compare 2 output enable  
Refer to CC1E description
- Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity  
CC1 channel configured as output:  
0: OC1N active high  
1: OC1N active low  
CC1 channel configured as input:  
This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.
- Note:* 1. This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S="00" (the channel is configured in output).  
2. On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.
- Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable  
0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.  
1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
- Bit 1 **CC1P**: Capture/Compare 1 output polarity  
CC1 channel configured as output:  
0: OC1 active high  
1: OC1 active low  
**CC1 channel configured as input:** The CC1NP/CC1P bits select the polarity of TI1FP1 and TI2FP1 for trigger or capture operations.  
00: non-inverted/rising edge. The circuit is sensitive to TlxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TlxFP1 is not inverted (trigger operation in gated mode).  
01: inverted/falling edge. The circuit is sensitive to TlxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TlxFP1 is inverted (trigger operation in gated mode).  
10: reserved, do not use this configuration.  
11: non-inverted/both edges. The circuit is sensitive to both TlxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TlxFP1 is not inverted (trigger operation in gated mode).
- Note:* 1. This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).  
2. On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

**CC1 channel configured as output:**

0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSI, OIS1, OIS1N and CC1NE bits.

1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSI, OIS1, OIS1N and CC1NE bits.

**CC1 channel configured as input:** This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx\_CCR1) or not.

0: Capture disabled

1: Capture enabled

**Table 227. Output control bits for complementary OCx and OCxN channels with break feature (TIM15)**

Control bits					Output states <sup>(1)</sup>	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) OCx=0 OCxN=0, OCxN_EN=0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) OCx=0	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		0	1	0	OCxREF + Polarity OCx=OCxREF XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) OCxN=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		1	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP, OCx_EN=1	Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1
0	0	X	X	X	Output disabled (not driven by the timer anymore). The output state is defined by the GPIO controller and can be High, Low or Hi-Z.	
	1		0	1	Off-State (output enabled with inactive state)	
			1	0	Asynchronously: OCx=CCxP, OCxN=CCxNP	
			1	1	Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state	
				1	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 AFIO registers.*

### 36.5.9 TIM15 counter (TIM15\_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit in the TIMx\_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

### 36.5.10 TIM15 prescaler (TIM15\_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK\_CNT) is equal to  $f_{CK\_PSC} / (PSC[15:0] + 1)$ .

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx\_EGR register or through trigger controller when configured in “reset mode”).

### 36.5.11 TIM15 auto-reload register (TIM15\_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the [Section 36.4.1: Time-base unit on page 1264](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.



### 36.5.12 TIM15 repetition counter register (TIM15\_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								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.

### 36.5.13 TIM15 capture/compare register 1 (TIM15\_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
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).

### 36.5.14 TIM15 capture/compare register 2 (TIM15\_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

**If channel CC2 is configured as output:**

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signalled on OC2 output.

**If channel CC2 is configured as input:**

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

### 36.5.15 TIM15 break and dead-time register (TIM15\_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	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, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx\_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional

- 0: Break input BRK in input mode
- 1: Break input BRK in bidirectional mode

In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

*Note:* This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.



Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm

- 0: Break input BRK is armed
- 1: Break input BRK is disarmed

This bit is cleared by hardware when no break source is active.

The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 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 OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx\_CCER register)

See OC/OCN enable description for more details ([Section 36.5.8: TIM15 capture/compare enable register \(TIM15\\_CCER\) on page 1311](#)).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

*Note: 1: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*2: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 12 **BKE**: Break enable

0: Break inputs (BRK and CCS clock failure event) disabled

1: Break inputs (BRK and CCS clock failure event) enabled

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details ([Section 36.5.8: TIM15 capture/compare enable register \(TIM15\\_CCER\) on page 1311](#)).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the AFIO logic, which forces a Hi-Z state)

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

See OC/OCN enable description for more details ([Section 36.5.8: TIM15 capture/compare enable register \(TIM15\\_CCER\) on page 1311](#)).

0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)

1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIMx\_BDTR register, OISx and OISxN bits in TIMx\_CR2 register and BKE/BKP/AOE bits in TIMx\_BDTR register can no longer be written

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx\_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx\_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

*Note: The LOCK bits can be written only once after the reset. Once the TIMx\_BDTR register has been written, their content is frozen until the next reset.*

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5]=0xx => DT=DTG[7:0]x  $t_{dtg}$  with  $t_{dtg}=t_{DTS}$

DTG[7:5]=10x => DT=(64+DTG[5:0])x $t_{dtg}$  with  $T_{dtg}=2xt_{DTS}$

DTG[7:5]=110 => DT=(32+DTG[4:0])x $t_{dtg}$  with  $T_{dtg}=8xt_{DTS}$

DTG[7:5]=111 => DT=(32+DTG[4:0])x $t_{dtg}$  with  $T_{dtg}=16xt_{DTS}$

Example if  $T_{DTS}=125\text{ns}$  (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16  $\mu\text{s}$  to 31750 ns by 250 ns steps,

32  $\mu\text{s}$  to 63  $\mu\text{s}$  by 1  $\mu\text{s}$  steps,

64  $\mu\text{s}$  to 126  $\mu\text{s}$  by 2  $\mu\text{s}$  steps

*Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

### 36.5.16 TIM15 DMA control register (TIM15\_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address).

- 00000: 1 transfer,
- 00001: 2 transfers,
- 00010: 3 transfers,
- ...
- 10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register.

Example:

- 00000: TIMx\_CR1,
- 00001: TIMx\_CR2,
- 00010: TIMx\_SMCR,
- ...

### 36.5.17 TIM15 DMA address for full transfer (TIM15\_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

$$(TIMx\_CR1 \text{ address}) + (DBA + \text{DMA index}) \times 4$$

where TIMx\_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx\_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx\_DCR).

### 36.5.18 TIM15 option register 1 (TIM15\_OR1)

Address offset: 0x50

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENCODER_MODE[1:0]	T11_RMP	
													rw	rw	rw

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

Bits 2:1 **ENCODER\_MODE[1:0]**: Encoder mode

00: No redirection

01:TIM2 IC1 and TIM2 IC2 are connected to TIM15 IC1 and TIM15 IC2 respectively

10:TIM3 IC1 and TIM3 IC2 are connected to TIM15 IC1 and TIM15 IC2 respectively

11:TIM4 IC1 and TIM4 IC2 are connected to TIM15 IC1 and TIM15 IC2 respectively

Bit 0 **T11\_RMP**: Input capture 1 remap

0: TIM15 input capture 1 is connected to I/O

1: TIM15 input capture 1 is connected to LSE

### 36.5.19 TIM15 option register 2 (TIM15\_OR2)

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	BKDF1 BK0E	Res.	Res.	Res.	Res.	Res.	BKCM P2E	BKCM P1E	BKINE
				rw	rw	rw	rw						rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BKCM P2P**: BRK COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP2 input is active low

1: COMP2 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 10 **BKCM P1P**: BRK COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP1 input is active low

1: COMP1 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

**Bit 9 BKINP:** BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input is active low

1: BKIN input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

**Bit 8 BKDF1BK0E:** BRK dfsdm1\_break[0] enable

This bit enables the dfsdm1\_break[0] for the timer's BRK input. dfsdm1\_break[0] output is 'ORed' with the other BRK sources.

0: dfsdm1\_break[0]input disabled

1: dfsdm1\_break[0]input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

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

**Bit 2 BKCOMP2E:** BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

0: COMP2 input disabled

1: COMP2 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

**Bit 1 BKCOMP1E:** BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

0: COMP1 input disabled

1: COMP1 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

**Bit 0 BKINE:** BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

### 36.5.20 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

**Table 228. TIM15 register map and reset values**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	TIM15_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]	ARPE	Res.	Res.	Res.	Res.	OPM	URS	UDIS	CEN	
	Reset value																					0		0	0	0				0	0	0	0	
0x04	TIM15_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	T1S	MMS[2:0]			CCDS	CCJUS	Res.	CCPC	
	Reset value																						0	0	0	0	0	0	0	0	0	0	0	0
0x08	TIM15_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]			
	Reset value															0										0	0	0	0		0	0	0	
0x0C	TIM15_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	Res.	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value																		0	0				0	0	0	0	0	0		0	0	0	
0x10	TIM15_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value																							0	0		0	0	0			0	0	0
0x14	TIM15_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG	
	Reset value																									0	0	0			0	0	0	
0x18	TIM15_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [2:0]	Res.	Res.	Res.	OC2PE	OC2FE	CC2S [1:0]	Res.	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]		
	Reset value							0											0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	TIM15_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]	Res.	Res.	IC2 PSC [1:0]	CC2S [1:0]	Res.	IC1F[3:0]			IC1 PSC [1:0]	CC1S [1:0]				
Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x20	TIM15_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																									0		0	0	0	0	0	0	









## 36.6 TIM16/TIM17 registers

Refer to [Section 1.2](#) for a list of abbreviations used in register descriptions.

### 36.6.1 TIM16/TIM17 control register 1 (TIMx\_CR1)

Address offset: 0x00

Reset value: 0x0000

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

### 36.6.2 TIM16/TIM17 control register 2 (TIMx\_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0

1: OC1N=1 after a dead-time when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BKR register).*

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BKR register).*

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

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

*Note: This bit acts only on channels that have a complementary output.*

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control  
 0: CCxE, CCxNE and OCxM bits are not preloaded  
 1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.

*Note: This bit acts only on channels that have a complementary output.*

### 36.6.3 TIM16/TIM17 DMA/interrupt enable register (TIMx\_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	COMDE	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
		rw				rw	rw	rw		rw				rw	rw

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **COMDE**: COM DMA request enable  
 0: COM DMA request disabled  
 1: COM DMA request enabled

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

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable  
 0: CC1 DMA request disabled  
 1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable  
 0: Update DMA request disabled  
 1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable  
 0: Break interrupt disabled  
 1: Break interrupt enabled

Bit 6 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

### 36.6.4 TIM16/TIM17 status register (TIMx\_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.

0: No overcapture has been detected

1: The counter value has been captured in TIMx\_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

**If channel CC1 is configured as output:**

This flag is set by hardware when the counter matches the compare value. It is cleared by software.

0: No match.

1: The content of the counter TIMx\_CNT matches the content of the TIMx\_CCR1 register.

When the contents of TIMx\_CCR1 are greater than the contents of TIMx\_ARR, the CC1IF bit goes high on the counter overflow

**If channel CC1 is configured as input:**

This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx\_CCR1 register.

0: No input capture occurred

1: The counter value has been captured in TIMx\_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx\_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx\_EGR register, if URS=0 and UDIS=0 in the TIMx\_CR1 register.

### 36.6.5 TIM16/TIM17 event generation register (TIMx\_EGR)

Address offset: 0x14

Reset value: 0x0000

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

### 36.6.6 TIM16/TIM17 capture/compare mode register 1 (TIMx\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
															Res.
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[2:0]	OC1PE	OC1FE	CC1S[1:0]			
									IC1F[3:0]	IC1PSC[1:0]					
								rw	rw	rw	rw	rw	rw	rw	rw

#### Output compare mode:

Bits 31:17 Reserved, always read as 0

Bit 16 **OC1M[3]**: Output Compare 1 mode (bit 3)

Bits 15:7 Reserved

Bits 6:4 **OC1M[2:0]**: Output Compare 1 mode (bits 2 to 0)

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx\_CCR1 and the counter TIMx\_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0011: Toggle - OC1REF toggles when TIMx\_CNT=TIMx\_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx\_CNT<TIMx\_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx\_CNT<TIMx\_CCR1 else active.

All other values: Reserved

*Note: 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S='00' (the channel is configured in output).*

*2: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.*



**Bit 3 OC1PE:** Output Compare 1 preload enable

0: Preload register on TIMx\_CCR1 disabled. TIMx\_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx\_CCR1 enabled. Read/Write operations access the preload register. TIMx\_CCR1 preload value is loaded in the active register at each update event.

*Note:* **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S='00' (the channel is configured in output).

**2:** The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx\_CR1 register). Else the behavior is not guaranteed.

**Bit 2 OC1FE:** Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.  
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

**Bits 1:0 CC1S:** Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note:* CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx\_CCER).

### Input capture mode

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

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample T11 input and the length of the digital filter applied to T11. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at  $f_{DTS}$
- 0001:  $f_{SAMPLING}=f_{CK\_INT}$ , N=2
- 0010:  $f_{SAMPLING}=f_{CK\_INT}$ , N=4
- 0011:  $f_{SAMPLING}=f_{CK\_INT}$ , N=8
- 0100:  $f_{SAMPLING}=f_{DTS}/2$ , N=
- 0101:  $f_{SAMPLING}=f_{DTS}/2$ , N=8
- 0110:  $f_{SAMPLING}=f_{DTS}/4$ , N=6
- 0111:  $f_{SAMPLING}=f_{DTS}/4$ , N=8
- 1000:  $f_{SAMPLING}=f_{DTS}/8$ , N=6
- 1001:  $f_{SAMPLING}=f_{DTS}/8$ , N=8
- 1010:  $f_{SAMPLING}=f_{DTS}/16$ , N=5
- 1011:  $f_{SAMPLING}=f_{DTS}/16$ , N=6
- 1100:  $f_{SAMPLING}=f_{DTS}/16$ , N=8
- 1101:  $f_{SAMPLING}=f_{DTS}/32$ , N=5
- 1110:  $f_{SAMPLING}=f_{DTS}/32$ , N=6
- 1111:  $f_{SAMPLING}=f_{DTS}/32$ , N=8

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx\_CCER register).

- 00: no prescaler, capture is done each time an edge is detected on the capture input.
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, IC1 is mapped on T11
- 10: CC1 channel is configured as input, IC1 is mapped on T12
- 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx\_CCER).*

### 36.6.7 TIM16/TIM17 capture/compare enable register (TIMx\_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												r/w	r/w	r/w	r/w

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

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

- 0: OC1N active high
- 1: OC1N active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to the description of CC1P.

*Note:* **1.** This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S="00" (the channel is configured in output).

**2.** On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

**CC1 channel configured as output:**

- 0: OC1 active high
- 1: OC1 active low

**CC1 channel configured as input:**

The CC1NP/CC1P bits select the polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

00: Non-inverted/rising edge. The circuit is sensitive to TlxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TlxFP1 is not inverted (trigger operation in gated mode).

01: Inverted/falling edge. The circuit is sensitive to TlxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TlxFP1 is inverted (trigger operation in gated mode).

10: Reserved, do not use this configuration.

1: Non-inverted/both edges. The circuit is sensitive to both TlxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TlxFP1 is not inverted (trigger operation in gated mode).

*Note:* **1.** This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).

**2.** On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

**CC1 channel configured as output:**

0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

**CC1 channel configured as input:**

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx\_CCR1) or not.

- 0: Capture disabled
- 1: Capture enabled

**Table 229. 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 anymore). The output state is defined by the GPIO controller and can be High, Low or Hi-Z.	
	1		0	0		
			0	1		
			1	0		
			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 AFIO registers.*

### 36.6.8 TIM16/TIM17 counter (TIMx\_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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



Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx\_ISR register. If the UIFREMAP bit in TIMx\_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

### 36.6.9 TIM16/TIM17 prescaler (TIMx\_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK\_CNT) is equal to  $f_{CK\_PSC} / (PSC[15:0] + 1)$ .

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx\_EGR register or through trigger controller when configured in “reset mode”).

### 36.6.10 TIM16/TIM17 auto-reload register (TIMx\_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 36.4.1: Time-base unit on page 1264](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 36.6.11 TIM16/TIM17 repetition counter register (TIMx\_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								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.

### 36.6.12 TIM16/TIM17 capture/compare register 1 (TIMx\_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
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).

### 36.6.13 TIM16/TIM17 break and dead-time register (TIMx\_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	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, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx\_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional

- 0: Break input BRK in input mode
- 1: Break input BRK in bidirectional mode

In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

*Note:* This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm

- 0: Break input BRK is armed
- 1: Break input BRK is disarmed

This bit is cleared by hardware when no break source is active.

The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 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 OSSI bit.
- 1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx\_CCER register)

See OC/OCN enable description for more details (Section 36.6.7: TIM16/TIM17 capture/compare enable register (TIMx\_CCER) on page 1332).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

*Note: 1. This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*2. Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 12 **BKE**: Break enable

0: Break inputs (BRK and CCS clock failure event) disabled

1: Break inputs (BRK and CCS clock failure event) enabled

*Note: 1. This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*2. Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details ([Section 36.6.7: TIM16/TIM17 capture/compare enable register \(TIMx\\_CCER\) on page 1332](#)).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the AFIO logic, which forces a Hi-Z state)

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx\_BDTR register).*



Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

See OC/OCN enable description for more details ([Section 36.6.7: TIM16/TIM17 capture/compare enable register \(TIMx\\_CCER\) on page 1332](#)).

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>=2x t<sub>DTS</sub>

DTG[7:5]=110 => DT=(32+DTG[4:0])x t<sub>dtg</sub> with T<sub>dtg</sub>=8x t<sub>DTS</sub>

DTG[7:5]=111 => DT=(32+DTG[4:0])x t<sub>dtg</sub> with T<sub>dtg</sub>=16x t<sub>DTS</sub>

Example if T<sub>DTS</sub>=125ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 μs to 31750 ns by 250 ns steps,

32 μs to 63 μs by 1 μs steps,

64 μs to 126 μs by 2 μs steps

*Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

### 36.6.14 TIM16/TIM17 DMA control register (TIMx\_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer,

00001: 2 transfers,

00010: 3 transfers,

...

10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register.

Example:

00000: TIMx\_CR1,

00001: TIMx\_CR2,

00010: TIMx\_SMCR,

...

**Example:** Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx\_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx\_CR1 address.

### 36.6.15 TIM16/TIM17 DMA address for full transfer (TIMx\_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address  
 $(\text{TIMx\_CR1 address}) + (\text{DBA} + \text{DMA index}) \times 4$

where TIMx\_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx\_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx\_DCR).

### 36.6.16 TIM16 option register 1 (TIM16\_OR1)

Address offset: 0x50

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits1:0 **TI1\_RMP[1:0]**: Input capture 1 remap

00: TIM16 input capture 1 is connected to I/O

01: TIM16 input capture 1 is connected to LSI

10: TIM16 input capture 1 is connected to LSE

11: TIM16 input capture 1 is connected to RTC wakeup interrupt

### 36.6.17 TIM16 option register 2 (TIM16\_OR2)

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	BKDF1 BK1E	Res.	Res.	Res.	Res.	Res.	BKCM P2E	BKCM P1E	BKINE
				rw	rw	rw	rw						rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BKCM P2P**: BRK COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP2 input is active low

1: COMP2 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 10 **BKCM P1P**: BRK COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP1 input is active low

1: COMP1 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

- 0: BKIN input is active low
- 1: BKIN input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 8 **BKDF1BK1E**: BRK dfsdm1\_break[1] enable

This bit enables the dfsdm1\_break[1] for the timer's BRK input. dfsdm1\_break[1] output is 'ORed' with the other BRK sources.

- 0: dfsdm1\_break[1] input disabled
- 1: dfsdm1\_break[1] input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

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

Bit 2 **BKCOMP2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

- 0: COMP2 input disabled
- 1: COMP2 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 1 **BKCOMP1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

- 0: COMP1 input disabled
- 1: COMP1 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

- 0: BKIN input disabled
- 1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

### 36.6.18 TIM17 option register 1 (TIM17\_OR1)

Address offset: 0x50

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T11_RMP[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

- Bits1:0 **TI1\_RMP[1:0]**: Input capture 1 remap
  - 00: TIM17 input capture 1 is connected to I/O
  - 01: TIM17 input capture 1 is connected to MSI
  - 10: TIM17 input capture 1 is connected to HSE/32
  - 11: TIM17 input capture 1 is connected to MCO

### 36.6.19 TIM17 option register 2 (TIM17\_OR2)

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	BKDF1 BK2E	Res.	Res.	Res.	Res.	Res.	BKCM P2E	BKCM P1E	BKINE
				rw	rw	rw	rw						rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

- Bit 11 **BKCOMP2P**: BRK COMP2 input polarity
  - This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.
  - 0: COMP2 input 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 **BKCOMP1P**: 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).*
- Bit 8 **BKDF1BK2E**: BRK dfsdm1\_break[2] enable
  - This bit enables the dfsdm1\_break[2] for the timer's BRK input. dfsdm1\_break[2] output is 'ORed' with the other BRK sources.
  - 0: dfsdm1\_break[2] input disabled
  - 1: dfsdm1\_break[2] input enabled
  - Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

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

Bit 2 **BKCMP2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

0: COMP2 input disabled

1: COMP2 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 1 **BKCMP1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

0: COMP1 input disabled

1: COMP1 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*



Table 230. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
0x30	TIMx_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]										
	Reset value																									0	0	0	0	0	0	0	0	0		
0x34	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x44	TIMx_BDTR	Res.	Res.	Res.	BKID	Res.	BKSRM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]	DT[7:0]											
	Reset value				0		0											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x48	TIMx_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]				DBA[4:0]											
	Reset value																				0	0	0	0	0					0	0	0	0			
0x4C	TIMx_DMAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x50	TIM16_OR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM16_OR1 [1:0]		
	Reset value																																	0	0	
0x60	TIM16_OR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKMP2P	BKMP1P	BKINP	BKDF1BK1E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKMP2E	BKMP1E	BKINE
	Reset value																					0	0	0	0									0	0	1
0x50	TIM17_OR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM17_OR1 [1:0]		
	Reset value																																		0	0
0x60	TIM17_OR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKMP2P	BKMP1P	BKINP	BKDF1BK2E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKMP2E	BKMP1E	BKINE
	Reset value																					0	0	0	0									0	0	1

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.





## 37 Basic timers (TIM6/TIM7)

### 37.1 TIM6/TIM7 introduction

The basic timers TIM6 and TIM7 consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time-base generation but they are also specifically used to drive the digital-to-analog converter (DAC). In fact, the timers are internally connected to the DAC and are able to drive it through their trigger outputs.

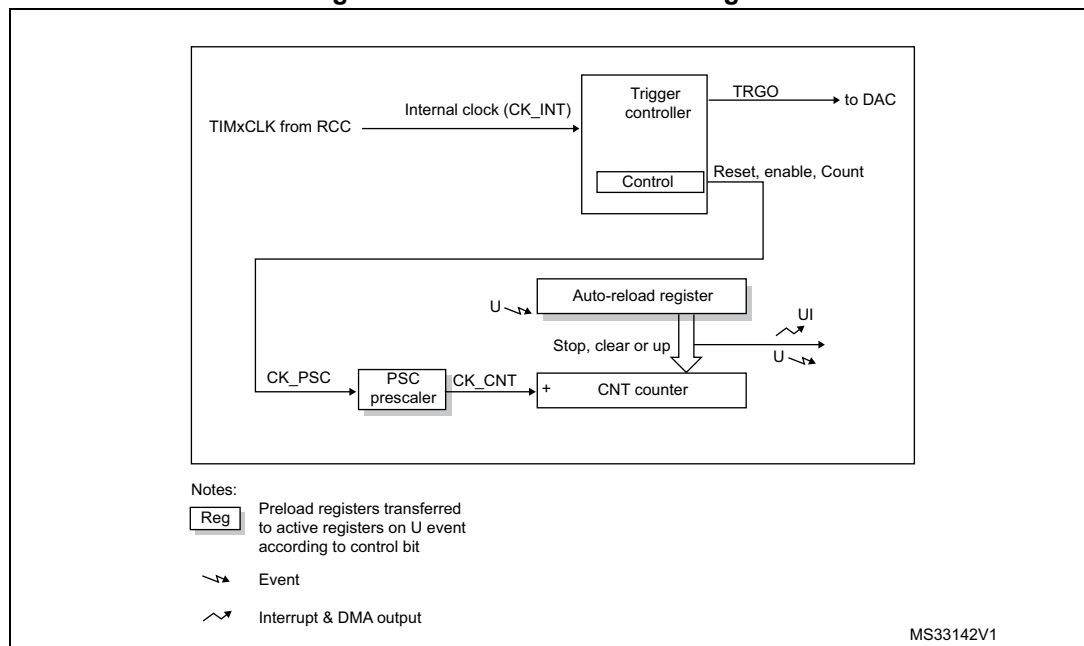
The timers are completely independent, and do not share any resources.

### 37.2 TIM6/TIM7 main features

Basic timer (TIM6/TIM7) features include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

Figure 399. Basic timer block diagram



## 37.3 TIM6/TIM7 functional description

### 37.3.1 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx\_CNT)
- Prescaler Register (TIMx\_PSC)
- Auto-Reload Register (TIMx\_ARR)

The auto-reload register is preloaded. The preload register is accessed each time an attempt is made to write or read the auto-reload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the auto-reload preload enable bit (ARPE) in the TIMx\_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx\_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in the TIMx\_CR1 register is set.

Note that the actual counter enable signal CNT\_EN is set 1 clock cycle after CEN.

#### Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx\_PSC register). It can be changed on the fly as the TIMx\_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 400* and *Figure 401* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 400. Counter timing diagram with prescaler division change from 1 to 2

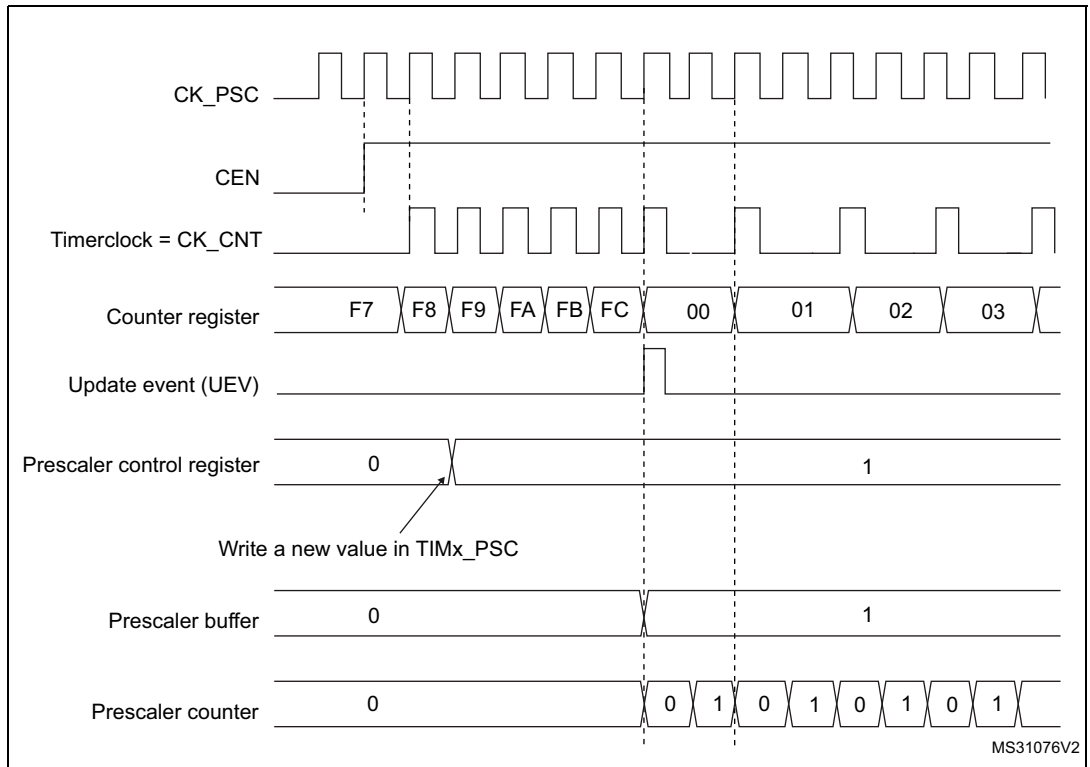
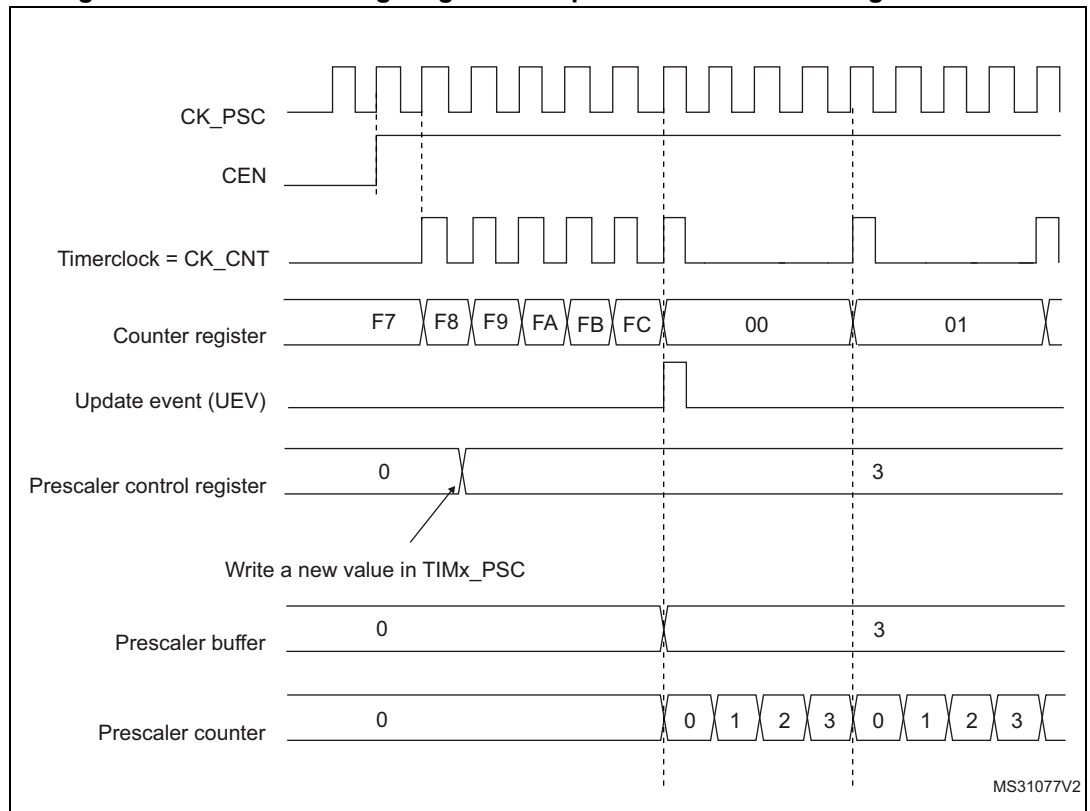


Figure 401. Counter timing diagram with prescaler division change from 1 to 4



### 37.3.2 Counting mode

The counter counts from 0 to the auto-reload value (contents of the TIMx\_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow or by setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx\_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx\_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx\_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx\_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx\_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR = 0x36.

**Figure 402. Counter timing diagram, internal clock divided by 1**

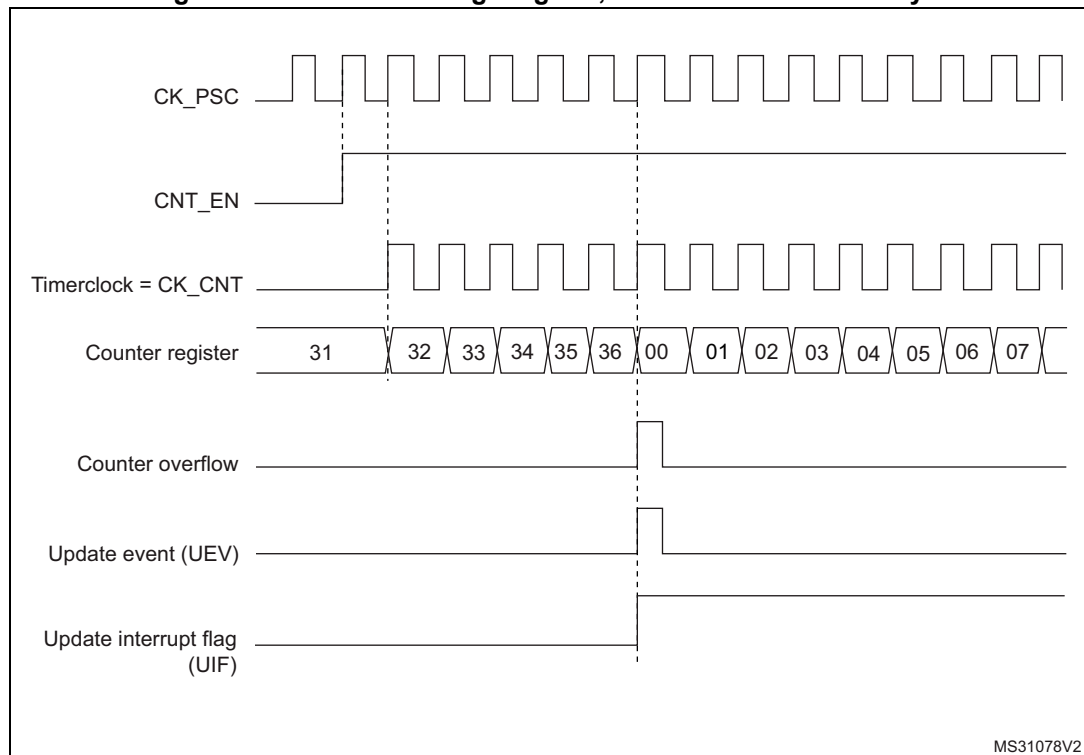


Figure 403. Counter timing diagram, internal clock divided by 2

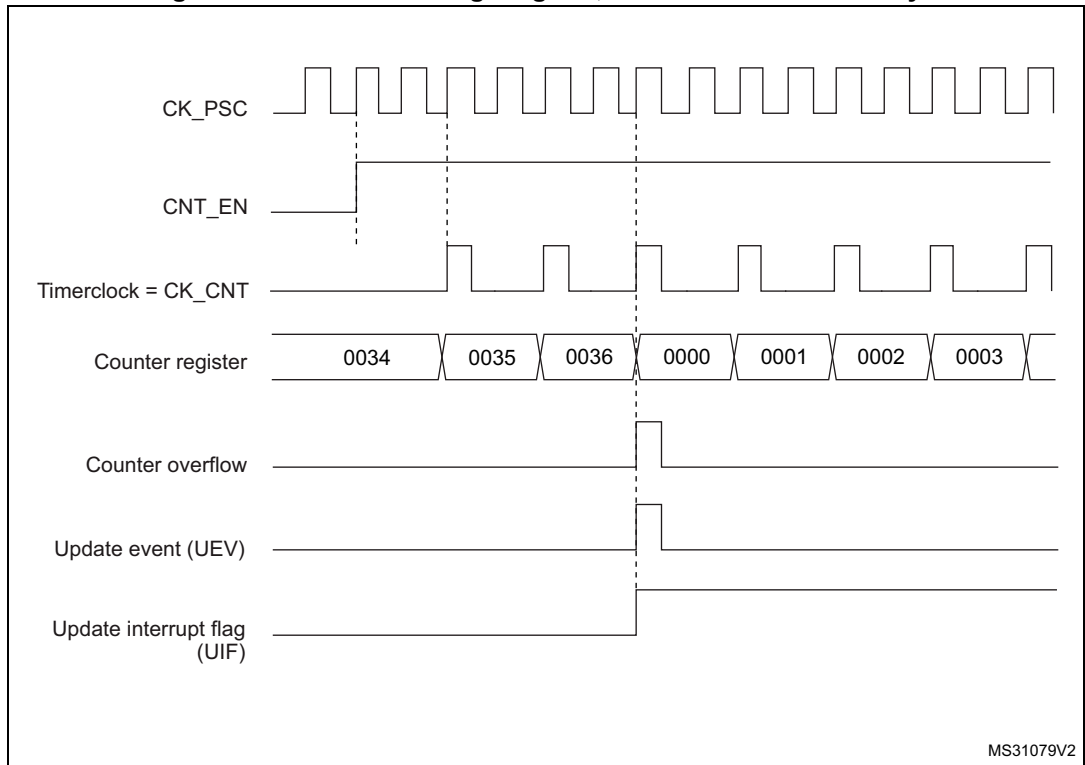


Figure 404. Counter timing diagram, internal clock divided by 4

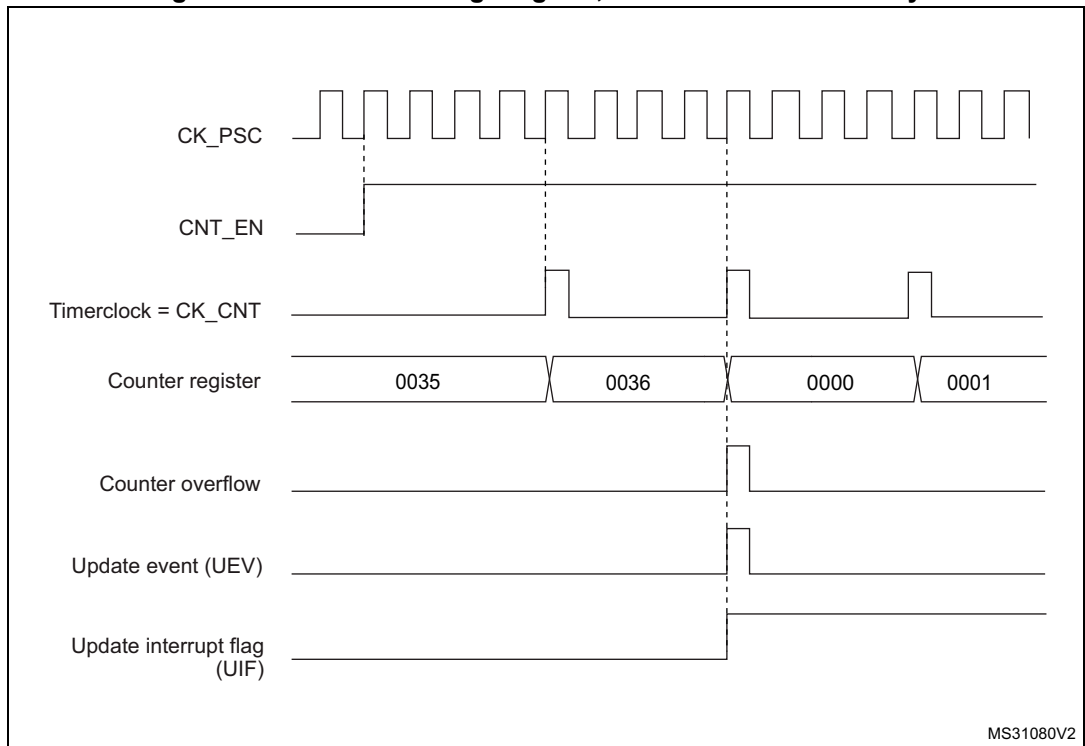
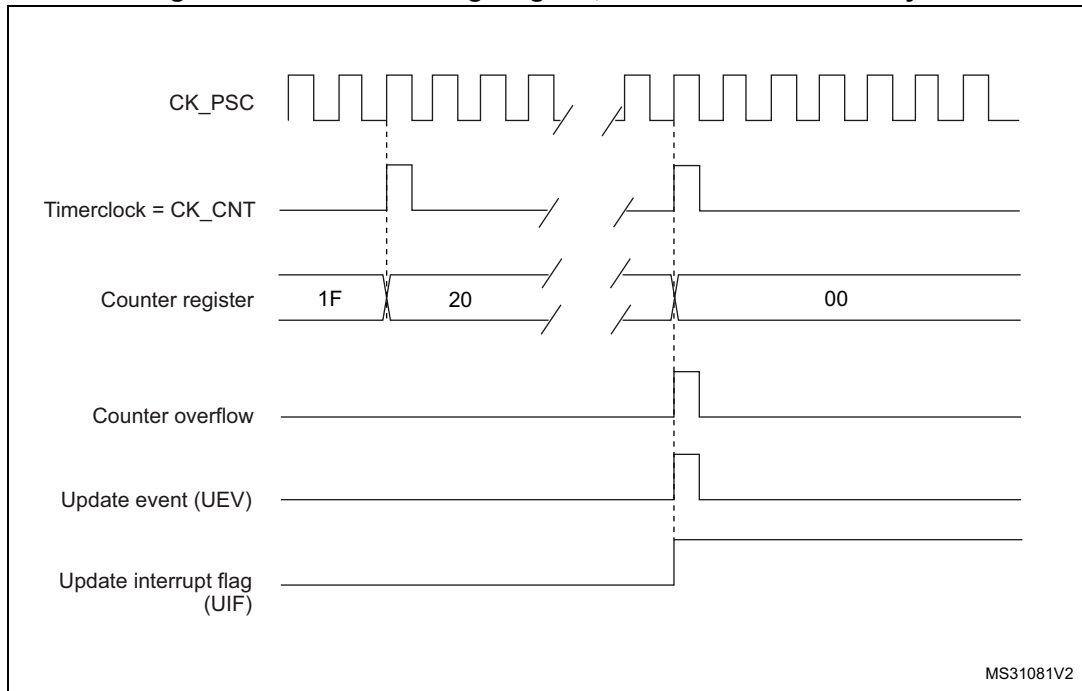
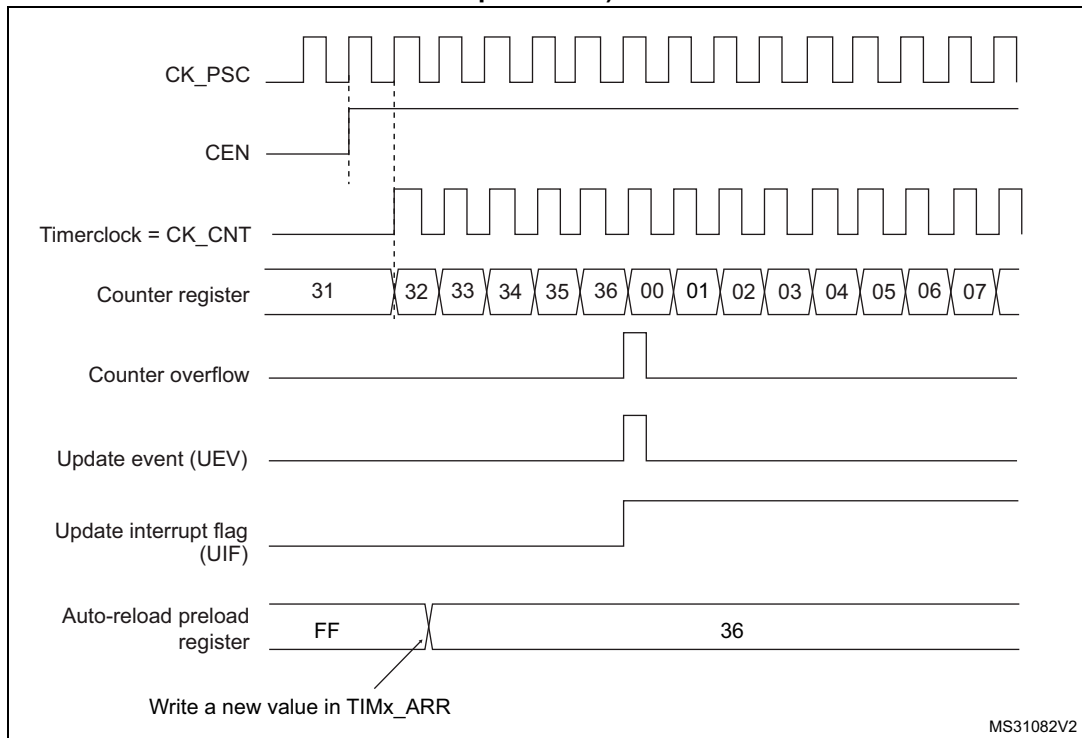


Figure 405. Counter timing diagram, internal clock divided by N



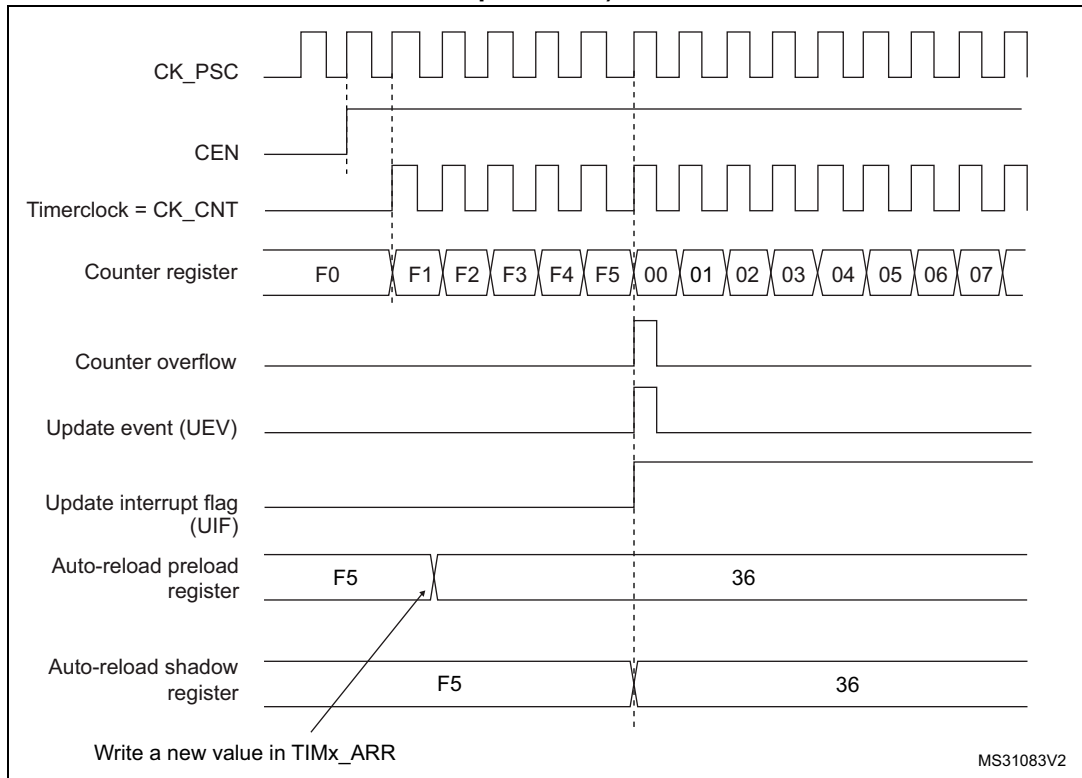
MS31081V2

Figure 406. Counter timing diagram, update event when ARPE = 0 (TIMx\_ARR not preloaded)



MS31082V2

**Figure 407. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)**



### 37.3.3 UIF bit remapping

The IUFREMAP bit in the TIMx\_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

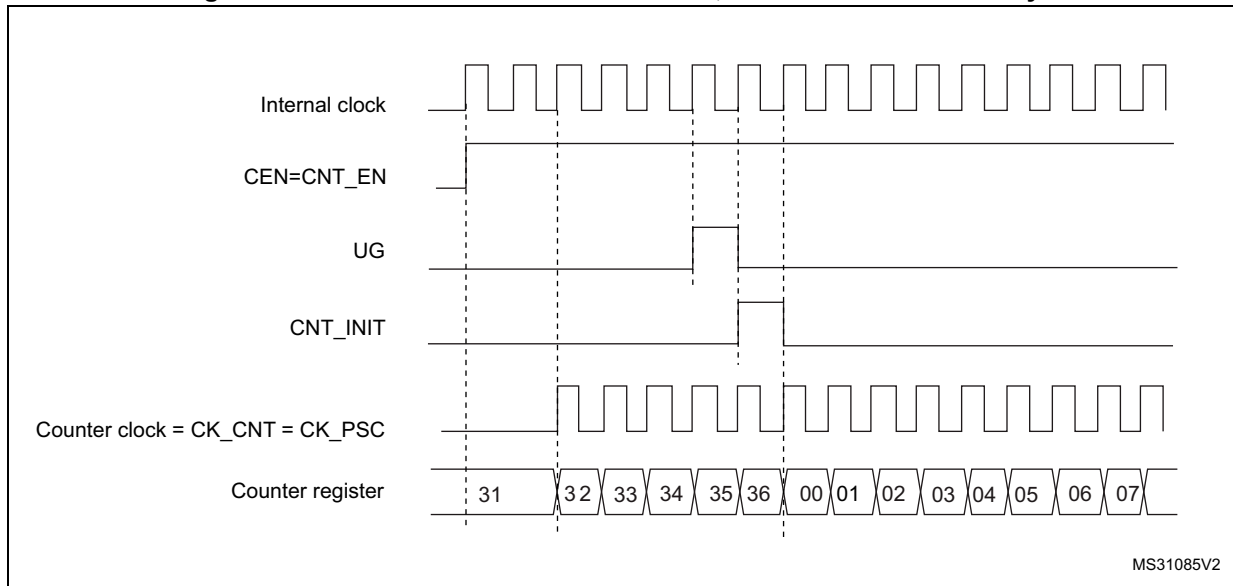
### 37.3.4 Clock source

The counter clock is provided by the Internal clock (CK\_INT) source.

The CEN (in the TIMx\_CR1 register) and UG bits (in the TIMx\_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK\_INT.

*Figure 408* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 408. Control circuit in normal mode, internal clock divided by 1



MS31085V2

### 37.3.5 Debug mode

When the microcontroller enters the debug mode (Cortex®-M4 core - halted), the TIMx counter either continues to work normally or stops, depending on the DBG\_TIMx\_STOP configuration bit in the DBG module. For more details, refer to [Section 46.16.2: Debug support for timers, RTC, watchdog, bxCAN and I2C](#).

## 37.4 TIM6/TIM7 registers

Refer to [Section 1.2 on page 75](#) for a list of abbreviations used in register descriptions.

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

### 37.4.1 TIM6/TIM7 control register 1 (TIMx\_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw				rw				rw	rw	rw	rw

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

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx\_CNT register bit 31.

1: Remapping enabled. UIF status bit is copied to TIMx\_CNT register bit 31.

Bits 10:8 Reserved, must be kept at reset value.



- Bit 7 **ARPE**: Auto-reload preload enable  
0: TIMx\_ARR register is not buffered.  
1: TIMx\_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

- Bit 3 **OPM**: One-pulse mode  
0: Counter is not stopped at update event  
1: Counter stops counting at the next update event (clearing the CEN bit).

- Bit 2 **URS**: Update request source  
This bit is set and cleared by software to select the UEV event sources.  
0: Any of the following events generates an update interrupt or DMA request if enabled.  
These events can be:
- Counter overflow/underflow
  - Setting the UG bit
  - Update generation through the slave mode controller
- 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

- Bit 1 **UDIS**: Update disable  
This bit is set and cleared by software to enable/disable UEV event generation.  
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
- Counter overflow/underflow
  - Setting the UG bit
  - Update generation through the slave mode controller
- Buffered registers are then loaded with their preload values.  
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

- Bit 0 **CEN**: Counter enable  
0: Counter disabled  
1: Counter enabled

*Note: Gated mode can work only if the CEN bit has been previously set by software.  
However trigger mode can set the CEN bit automatically by hardware.*

CEN is cleared automatically in one-pulse mode, when an update event occurs.

### 37.4.2 TIM6/TIM7 control register 2 (TIMx\_CR2)

Address offset: 0x04

Reset value: 0x0000

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

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

Bits 6:4 **MMS**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx\_EGR register is used as a trigger output (TRGO). If reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter enable signal, CNT\_EN, is used as a trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIMx\_SMCR register).

010: **Update** - The update event is selected as a trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

*Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.*

Bits 3:0 Reserved, must be kept at reset value.

### 37.4.3 TIM6/TIM7 DMA/Interrupt enable register (TIMx\_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
							rw								rw

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled.

1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled.

1: Update interrupt enabled.

### 37.4.4 TIM6/TIM7 status register (TIMx\_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow or underflow regarding the repetition counter value and if UDIS = 0 in the TIMx\_CR1 register.
- When CNT is reinitialized by software using the UG bit in the TIMx\_EGR register, if URS = 0 and UDIS = 0 in the TIMx\_CR1 register.

### 37.4.5 TIM6/TIM7 event generation register (TIMx\_EGR)

Address offset: 0x14

Reset value: 0x0000

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

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

### 37.4.6 TIM6/TIM7 counter (TIMx\_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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



Bit 31 **UIFCPY**: UIF Copy  
 This bit is a read-only copy of the UIF bit of the TIMx\_ISR register. If the UIFREMAP bit in TIMx\_CR1 is reset, bit 31 is reserved and read as 0.

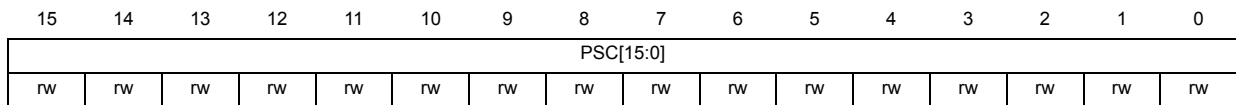
Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

### 37.4.7 TIM6/TIM7 prescaler (TIMx\_PSC)

Address offset: 0x28

Reset value: 0x0000

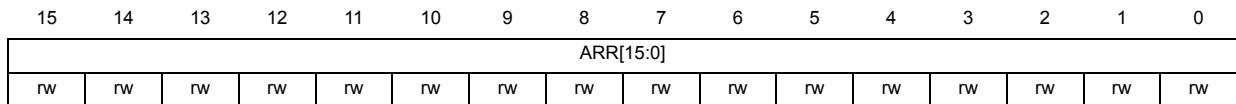


Bits 15:0 **PSC[15:0]**: Prescaler value  
 The counter clock frequency CK\_CNT is equal to  $f_{CK\_PSC} / (PSC[15:0] + 1)$ .  
 PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx\_EGR register or through trigger controller when configured in “reset mode”).

### 37.4.8 TIM6/TIM7 auto-reload register (TIMx\_ARR)

Address offset: 0x2C

Reset value: 0xFFFF



Bits 15:0 **ARR[15:0]**: Prescaler value  
 ARR is the value to be loaded into the actual auto-reload register.  
 Refer to [Section 37.3.1: Time-base unit on page 1348](#) for more details about ARR update and behavior.  
 The counter is blocked while the auto-reload value is null.

### 37.4.9 TIM6/TIM7 register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

**Table 231. TIM6/TIM7 register map and reset values**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0x00	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UJFREMA	Res.	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN	
	Reset value																						0				0				0	0	0	0	
0x04	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS [2:0]			Res.	Res.	Res.	Res.		
	Reset value																											0	0	0					
0x08	Reserved																																		
0x0C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																		0
0x10	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																		
0x14	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																		
0x18-0x20	Reserved																																		
0x24	TIMx_CNT	UJFCPY or Res.		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value	0																																	
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																		
0x2C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																		

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.



## 38 Low-power timer (LPTIM)

### 38.1 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.

### 38.2 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: LSE, LSI, HSI16 or APB clock
  - External clock source over LPTIM input (working with no LP 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

### 38.3 LPTIM implementation

[Table 232](#) describes LPTIM implementation on STM32L4Rxxx and STM32L4Sxxx devices: the full set of features is implemented in LPTIM1. LPTIM2 supports a smaller set of features, but is otherwise identical to LPTIM1.

**Table 232. STM32L4Rxxx and STM32L4Sxxx LPTIM features**

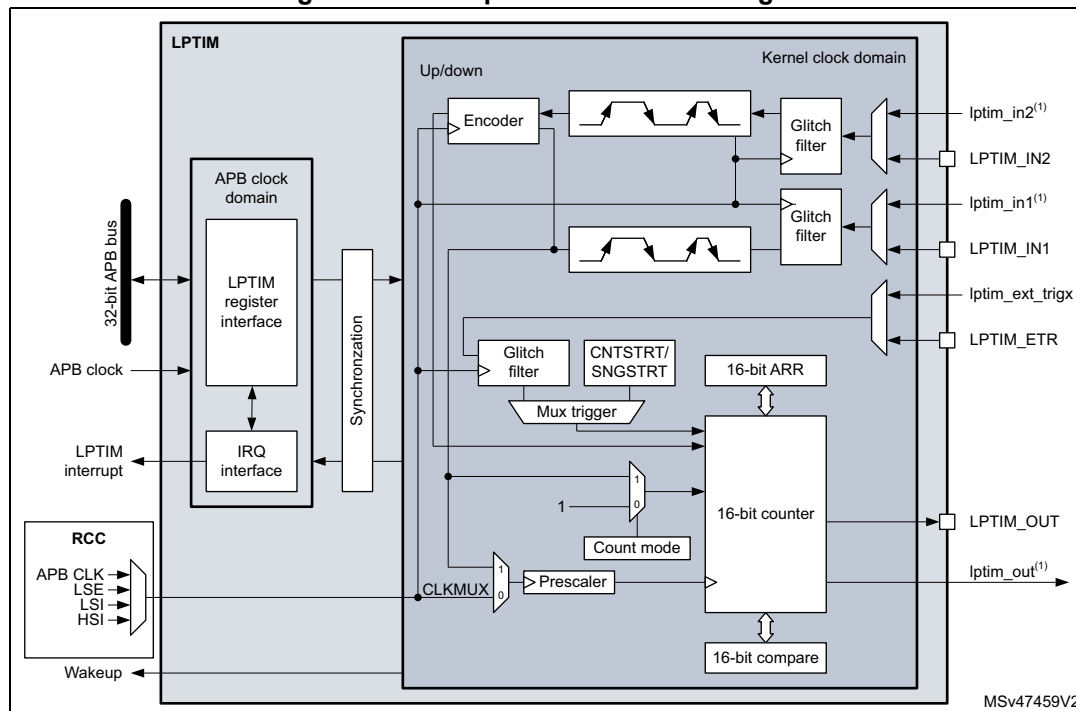
LPTIM modes/features <sup>(1)</sup>	LPTIM1	LPTIM2
Encoder mode	X	-

1. X = supported.

### 38.4 LPTIM functional description

#### 38.4.1 LPTIM block diagram

Figure 409. 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.

#### 38.4.2 LPTIM trigger mapping

The LPTIM external trigger connections are detailed hereafter:

Table 233. LPTIM1 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

Table 234. 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

### 38.4.3 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 chosen among APB, LSI, LSE or HSI16 sources through the Reset and Clock controller (RCC). 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 either from APB or any other embedded oscillator including LSE, LSI and HSI16.
- 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.

### 38.4.4 Glitch filter

The LPTIM inputs, either external (mapped to microcontroller GPIOs) or internal (mapped on the chip-level to other embedded peripherals, such as embedded comparators), 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.



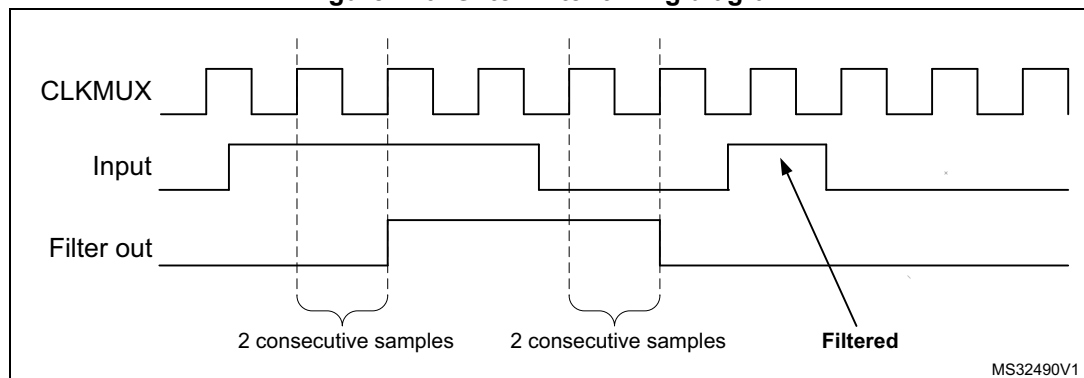
The digital filters are divided into two groups:

- 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 410](#) shows an example of glitch filter behavior in case of a 2 consecutive samples programmed.

**Figure 410. 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.*

### 38.4.5 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 235. Prescaler division ratios**

programming	dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

### 38.4.6 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.*

### 38.4.7 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.

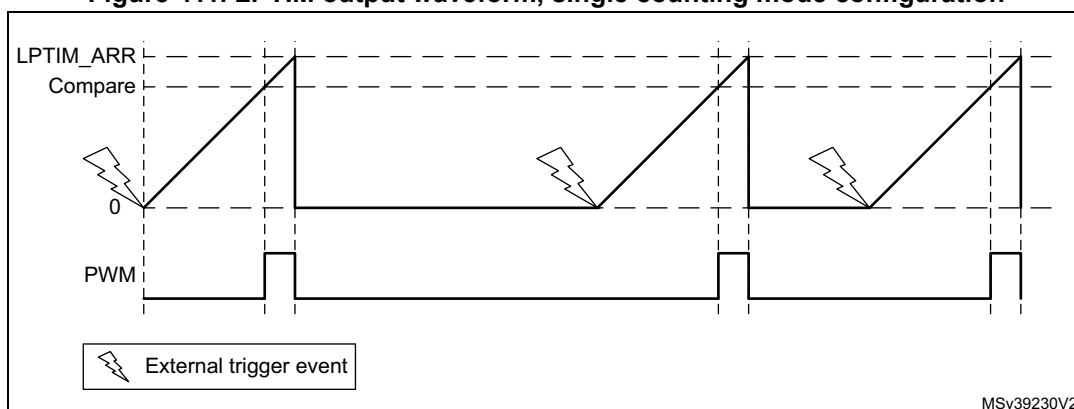
#### One-shot mode

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 411](#).

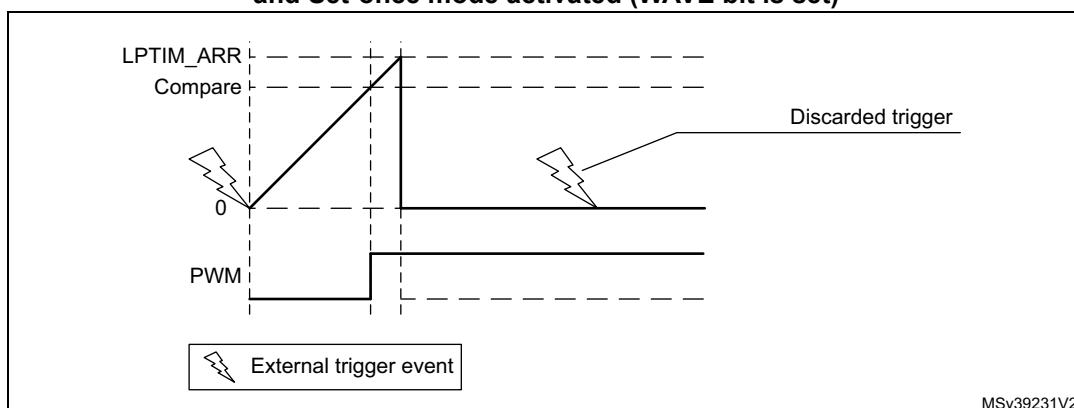
**Figure 411. LPTIM output waveform, single counting mode configuration**



- 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 412](#).

**Figure 412. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set)**



In case of software start (TRIGEN[1:0] = '00'), the SNGSTRT setting will start the counter for one-shot counting.

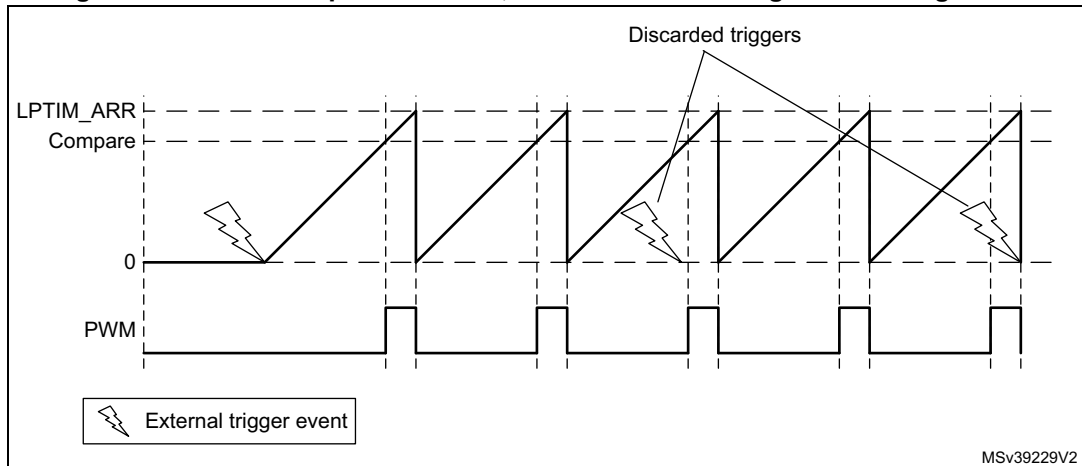
**Continuous mode**

To enable the continuous counting, the CNTSTRT bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set will start the counter for continuous counting. Any subsequent external trigger event will be discarded as shown in [Figure 413](#).

In case of software start (TRIGEN[1:0] = '00'), setting CNTSTRT will start the counter for continuous counting.

Figure 413. 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.

### 38.4.8 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 TIMOUT 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.

### 38.4.9 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 a match occurs between the LPTIM\_CMP and the LPTIM\_CNT registers. The LPTIM output is reset as soon as a match occurs between the LPTIM\_ARR and the LPTIM\_CNT registers
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require that the LPTIM\_ARR register value be strictly greater than the LPTIM\_CMP register value.

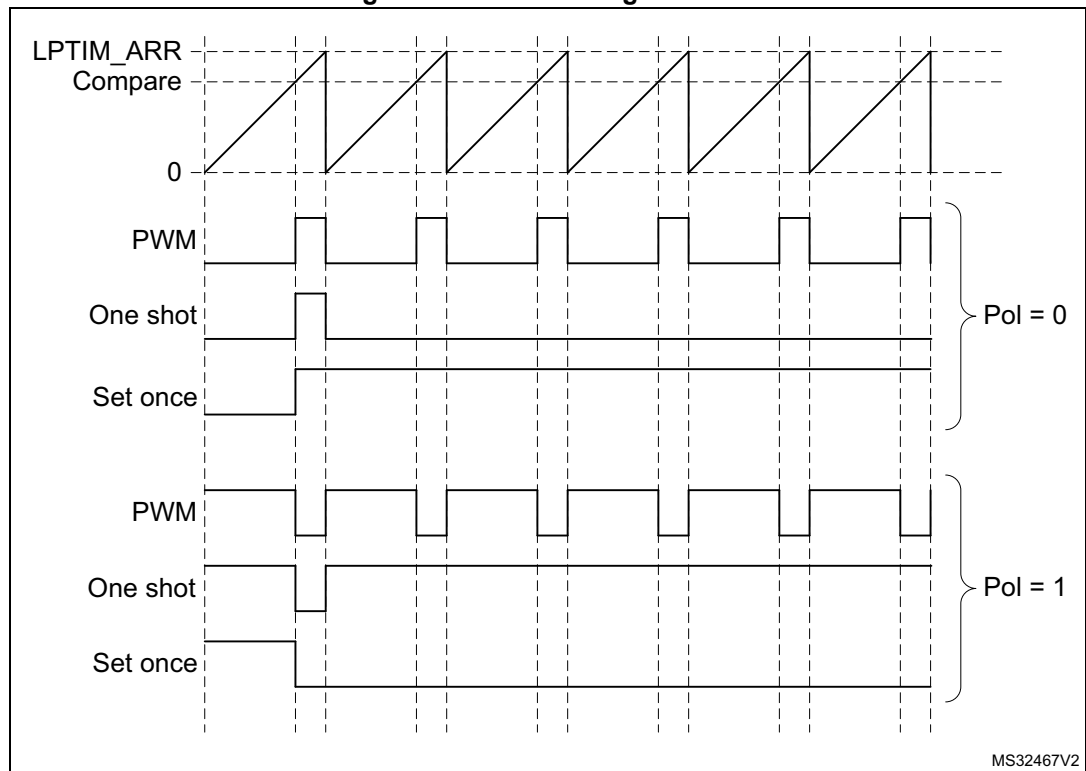
The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to '0' forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to '1' forces the LPTIM to generate a Set-once mode waveform.

The WAVPOL bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value will change immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by 2 can be generated. [Figure 414](#) 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 414. Waveform generation**



### 38.4.10 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.

### 38.4.11 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.

### 38.4.12 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.

### 38.4.13 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.

---

### 38.4.14 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 you must configure LPTIM\_ARR before starting. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signaled by the two Down and Up flags in the LPTIM\_ISR register. Also, an interrupt can be generated for both direction change events if enabled through the LPTIM\_IER register.

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 236. Encoder counting scenarios**

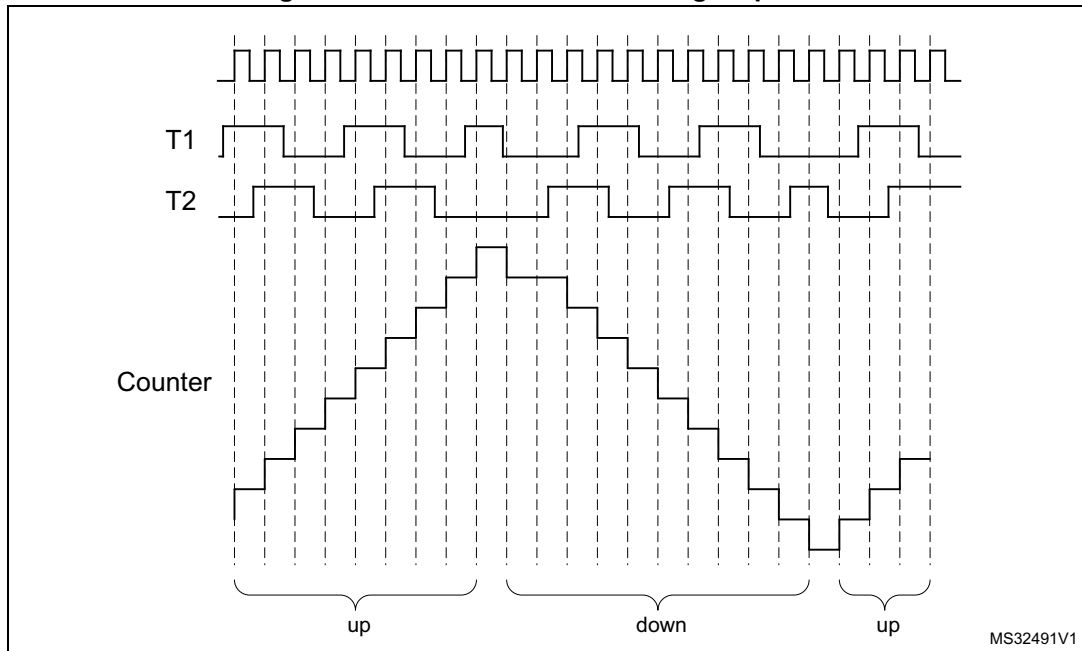
Active edge	Level on opposite signal (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
Rising Edge	High	Down	No count	Up	No count
	Low	Up	No count	Down	No count
Falling Edge	High	No count	Up	No count	Down
	Low	No count	Down	No count	Up
Both Edges	High	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 415. Encoder mode counting sequence



### 38.5 LPTIM low power modes

Table 237. Effect of low-power modes on the LPTIM

Mode	Description
Sleep	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Low-power run	No effect.
Low-power sleep	No effect. LPTIM interrupts cause the device to exit the Low-power sleep mode.
Stop 0 / Stop 1	No effect when LPTIM is clocked by LSE or LSI. LPTIM interrupts cause the device to exit Stop 0 and Stop 1.
Stop 2	No effect on LPTIM1 when LPTIM1 is clocked by LSE or LSI. LPTIM1 interrupts cause the device to exit Stop 2. LPTIM2 registers content is kept but LPTIM2 must be disabled before entering Stop 2.
Standby	The LPTIM peripheral is powered down and must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	

### 38.6 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 238. 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.

### 38.7 LPTIM registers

#### 38.7.1 LPTIM interrupt and status register (LPTIM\_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.	Res.	Res.	Res.	Res.	DOWN	UP	ARROK	CMPOK	EXTTRIG	ARRM	CMPM
									r	r	r	r	r	r	r



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

Bits 8:7 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.

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.

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.

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.

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.

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.

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.

### 38.7.2 LPTIM interrupt clear register (LPTIM\_ICR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWN CF	UPCF	ARRO KCF	CMPO KCF	EXTTR IGCF	ARRM CF	CMPM CF
									w	w	w	w	w	w	w

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

Bits 8:7 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 LPT\_ISR register.

- Bit 5 **UPCF**: Direction change to UP Clear Flag  
Writing 1 to this bit clear the UP flag in the LPT\_ISR register.
- Bit 4 **ARROKCF**: Autoreload register update OK Clear Flag  
Writing 1 to this bit clears the ARROK flag in the LPT\_ISR register
- Bit 3 **CMPOKCF**: Compare register update OK Clear Flag  
Writing 1 to this bit clears the CMPOK flag in the LPT\_ISR register
- Bit 2 **EXTTRIGCF**: External trigger valid edge Clear Flag  
Writing 1 to this bit clears the EXTTRIG flag in the LPT\_ISR register
- Bit 1 **ARRMCF**: Autoreload match Clear Flag  
Writing 1 to this bit clears the ARRM flag in the LPT\_ISR register
- Bit 0 **CMPMCF**: compare match Clear Flag  
Writing 1 to this bit clears the CMP flag in the LPT\_ISR register

### 38.7.3 LPTIM interrupt enable register (LPTIM\_IER)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWNI E	UPIE	ARRO KIE	CMPO KIE	EXTTR IGIE	ARRMI E	CMPMI E
									rw	rw	rw	rw	rw	rw	rw

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

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

- Bit 6 **DOWNIE**: Direction change to down Interrupt Enable  
0: DOWN interrupt disabled  
1: DOWN interrupt enabled

- Bit 5 **UPIE**: Direction change to UP Interrupt Enable  
0: UP interrupt disabled  
1: UP interrupt enabled

- 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')

### 38.7.4 LPTIM configuration register (LPTIM\_CFGR)

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.	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

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\_ARR and LPTIM\_CMP registers  
 1: The LPTIM output reflects the inverse of the compare results between LPTIM\_ARR and LPTIM\_CMP registers

- Bit 20 **WAVE**: Waveform shape  
The WAVE bit controls the output shape  
0: Deactivate Set-once mode, PWM / One Pulse waveform (depending on OPMODE bit)  
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 38.4.2: 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 TRGFLT[1:0]:** Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: any trigger active level change is considered as a valid trigger

01: trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.

10: trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.

11: trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

**Bits 4:3 CKFLT[1:0]:** Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: any external clock signal level change is considered as a valid transition

01: external clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: external clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: external clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

**Bits 2:1 CKPOL[1:0]:** Clock Polarity

If LPTIM is clocked by an external clock source:

When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:

00: the rising edge is the active edge used for counting

01: the falling edge is the active edge used for counting

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.

11: not allowed

If the LPTIM is configured in Encoder mode (ENC bit is set):

00: the encoder sub-mode 1 is active

01: the encoder sub-mode 2 is active

10: the encoder sub-mode 3 is active

Refer to [Section 38.4.14: 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\_CFGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to '0').

### 38.7.5 LPTIM control register (LPTIM\_CR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST ARE	COUN TRST	CNT STRT	SNG STRT	ENA BLE
											w	rs	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RSTARE**: Reset after read enable

This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM\_CNT register will asynchronously reset LPTIM\_CNT register content.

**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).

**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

### 38.7.6 LPTIM compare register (LPTIM\_CMP)

Address offset: 0x14

Reset value: 0x0000 0000

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

### 38.7.7 LPTIM autoreload register (LPTIM\_ARR)

Address offset: 0x18

Reset value: 0x0000 0001

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

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

Bits 15:0 **ARR[15:0]**: Auto reload value

ARR is the autoreload value for the LPTIM.

This value must be strictly greater than the CMP[15:0] value.

**Caution:** The LPTIM\_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to '1').

### 38.7.8 LPTIM counter register (LPTIM\_CNT)

Address offset: 0x1C

Reset value: 0x0000 0000

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

### 38.7.9 LPTIM1 option register (LPTIM1\_OR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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

### 38.7.10 LPTIM2 option register (LPTIM2\_OR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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

*Note:* When both OR\_1 and OR\_0 are set, LPTIM2 input 1 is connected to (COMP1\_OUT OR COMP2\_OUT).

### 38.7.11 LPTIM register map

The following table summarizes the LPTIM registers.

**Table 239. LPTIM register map and reset values**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	LPTIM_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWN	UP	AROK	CMPOK	EXTTRIG	ARRM	CMPM		
	Reset value																										0	0	0	0	0	0	0	
0x04	LPTIM_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWNCF	UPCF	ARROKCF	CMPOKCF	EXTTRIGCF	ARRMCF	CMPMCF		
	Reset value																										0	0	0	0	0	0	0	
0x08	LPTIM_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWNIE	UPIE	ARROKIE	CMPOKIE	EXTTRIGIE	ARRMIE	CMPMIE		
	Reset value																										0	0	0	0	0	0	0	
0x0C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNTMODE	PRELOAD	WAVPOL	WAVE	TIMOUT	TRIGEN				TRIGSEL[2:0]								TRGFLT			CKFLT	CKPOL	CKSEL		
	Reset value								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	LPTIM_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE	
	Reset value																												0	0	0	0	0	
0x14	LPTIM_CMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																	
0x18	LPTIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																	
0x1C	LPTIM_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																	
0x20	LPTIM_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																	



Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.

## 39 Infrared interface (IRTIM)

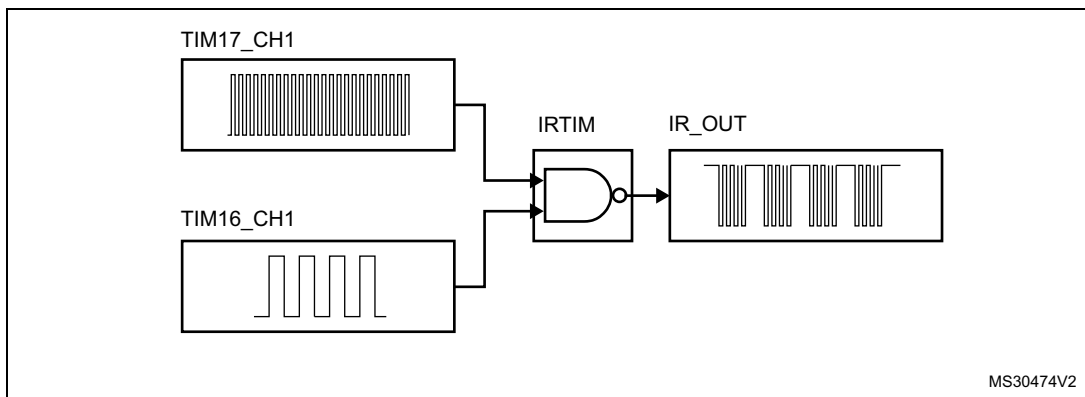
An infrared interface (IRTIM) for remote control is available on the device. It can be used with an infrared LED to perform remote control functions.

It uses internal connections with TIM16 and TIM17 as shown in [Figure 416](#).

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 416. 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.

## 40 Independent watchdog (IWDG)

### 40.1 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 41 on page 1394](#).

### 40.2 IWDG main features

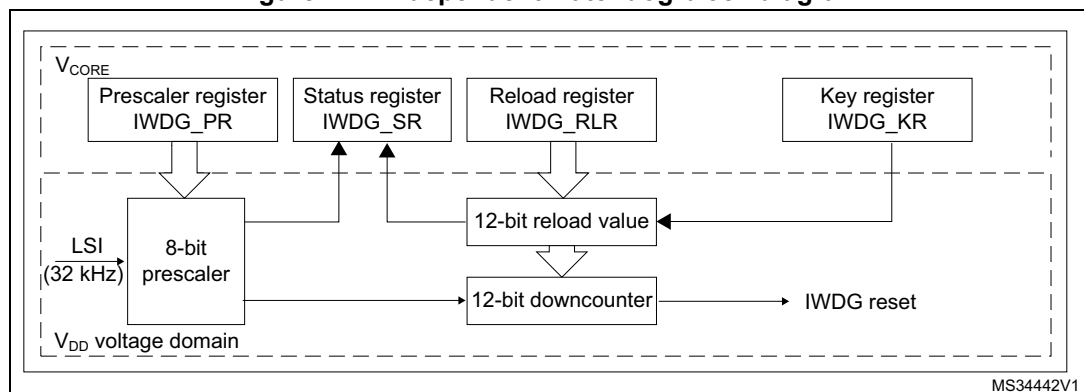
- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Conditional Reset
  - Reset (if watchdog activated) when the downcounter value becomes lower than 0x000
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window

### 40.3 IWDG functional description

#### 40.3.1 IWDG block diagram

[Figure 417](#) shows the functional blocks of the independent watchdog module.

**Figure 417. Independent watchdog block diagram**



1. The watchdog function is implemented in the  $V_{CORE}$  voltage domain that is still functional in Stop and Standby modes.

When the independent watchdog is started by writing the value 0x0000 CCCC in the *Key register (IWDG\_KR)*, the counter starts counting down from the reset value of 0xFFFF. When it reaches the end of count value (0x0000) a reset signal is generated (IWDG reset).

Whenever the key value 0x0000 AAAA is written in the *Key register (IWDG\_KR)*, the IWDG\_RLR value is reloaded in the counter and the watchdog reset is prevented.

### 40.3.2 Window option

The IWDG can also work as a window watchdog by setting the appropriate window in the *Window register (IWDG\_WINR)*.

If the reload operation is performed while the counter is greater than the value stored in the *Window register (IWDG\_WINR)*, then a reset is provided.

The default value of the *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 *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 *Key register (IWDG\_KR)*.
2. Enable register access by writing 0x0000 5555 in the *Key register (IWDG\_KR)*.
3. Write the IWDG prescaler by programming *Prescaler register (IWDG\_PR)* from 0 to 7.
4. Write the *Reload register (IWDG\_RLR)*.
5. Wait for the registers to be updated (IWDG\_SR = 0x0000 0000).
6. Write to the *Window register (IWDG\_WINR)*. This automatically refreshes the counter value in the *Reload register (IWDG\_RLR)*.

*Note:* Writing the window value allows to refresh the Counter value by the RLR when *Status register (IWDG\_SR)* is set to 0x0000 0000.

#### Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *Key register (IWDG\_KR)*.
2. Enable register access by writing 0x0000 5555 in the *Key register (IWDG\_KR)*.
3. Write the prescaler by programming the *Prescaler register (IWDG\_PR)* from 0 to 7.
4. Write the *Reload register (IWDG\_RLR)*.
5. Wait for the registers to be updated (IWDG\_SR = 0x0000 0000).
6. Refresh the counter value with IWDG\_RLR (IWDG\_KR = 0x0000 AAAA).



### 40.3.3 Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and generates a reset unless the *Key register (IWDG\_KR)* is written by the software before the counter reaches end of count or if the downcounter is reloaded inside the window.

### 40.3.4 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 *Section : User and read protection option bytes* for more details.

### 40.3.5 Behavior in Stop and Standby modes

Once running, the IWDG cannot be stopped.

### 40.3.6 Register access protection

Write access to *Prescaler register (IWDG\_PR)*, *Reload register (IWDG\_RLR)* and *Window register (IWDG\_WINR)* is protected. To modify them, the user must first write the code 0x0000 5555 in the *Key register (IWDG\_KR)*. A write access to this register with a different value will break the sequence and register access will be 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 the down-counter reload value or the window value is on going.

### 40.3.7 Debug mode

When the microcontroller enters Debug mode (core halted), the IWDG counter either continues to work normally or stops, depending on DBG\_IWDG\_STOP configuration bit in DBG module.

## 40.4 IWDG registers

Refer to [Section 1.2 on page 75](#) for a list of abbreviations used in register descriptions.  
 The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

### 40.4.1 Key register (IWDG\_KR)

Address offset: 0x00

Reset value: 0x0000 0000 (reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]**: Key value (write only, read 0x0000)

These bits must be written by software at regular intervals with the key value 0xAAAA, otherwise the watchdog generates a reset when the counter reaches 0.

Writing the key value 0x5555 to enable access to the IWDG\_PR, IWDG\_RLR and IWDG\_WINR registers (see [Section 40.3.6: Register access protection](#))

Writing the key value 0xCCCC starts the watchdog (except if the hardware watchdog option is selected)

### 40.4.2 Prescaler register (IWDG\_PR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **PR[2:0]**: Prescaler divider

These bits are write access protected see [Section 40.3.6: Register access protection](#). They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the [Status register \(IWDG\\_SR\)](#) must be reset in order to be able to change the prescaler divider.

- 000: divider /4
- 001: divider /8
- 010: divider /16
- 011: divider /32
- 100: divider /64
- 101: divider /128
- 110: divider /256
- 111: divider /256

*Note: Reading this register returns the prescaler value from the V<sub>DD</sub> voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the [Status register \(IWDG\\_SR\)](#) is reset.*

### 40.4.3 Reload register (IWDG\_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF (reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RL[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

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 [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 [Status register \(IWDG\\_SR\)](#) must be reset to be able to change the reload value.

*Note: Reading this register returns the reload value from the V<sub>DD</sub> voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on it. For this reason the value read from this register is valid only when the RVU bit in the [Status register \(IWDG\\_SR\)](#) is reset.*

### 40.4.4 Status register (IWDG\_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (not reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WVU	RVU	PVU
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

**Bit 2 WVU:** Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V<sub>DD</sub> voltage domain (takes up to five LSI/Prescaler clock cycles).

Window value can be updated only when WVU bit is reset.

This bit is generated only if generic “window” = 1

**Bit 1 RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V<sub>DD</sub> voltage domain (takes up to five LSI/Prescaler clock cycles).

Reload value can be updated only when RVU bit is reset.

**Bit 0 PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V<sub>DD</sub> voltage domain (takes up to five LSI/Prescaler clock cycles).

Prescaler value can be updated only when PVU bit is reset.

*Note: If several reload, prescaler, or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, and to wait until WVU bit is reset before changing the window value. However, after updating the prescaler and/or the reload/window value it is not necessary to wait until RVU or PVU or WVU is reset before continuing code execution except in case of low-power mode entry.*

### 40.4.5 Window register (IWDG\_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF (reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WIN[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]**: Watchdog counter window value

These bits are write access protected, see [Section 40.3.6](#), 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 [Status register \(IWDG\\_SR\)](#) must be reset in order to be able to change the reload value.

*Note: Reading this register returns the reload value from the V<sub>DD</sub> voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the [Status register \(IWDG\\_SR\)](#) is reset.*

### 40.4.6 IWDG register map

The following table gives the IWDG register map and reset values.

**Table 240. IWDG register map and reset values**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0x00	IWDG_KR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	KEY[15:0]																	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x04	IWDG_PR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PR[2:0]		
	Reset value																																0	0	0
0x08	IWDG_RLR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RL[11:0]												
	Reset value																						1	1	1	1	1	1	1	1	1	1	1	1	
0x0C	IWDG_SR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	
	Reset value																																	0	0
0x10	IWDG_WINR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	WIN[11:0]												
	Reset value																						1	1	1	1	1	1	1	1	1	1	1	1	

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.

## 41 System window watchdog (WWDG)

### 41.1 Introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the downcounter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter has reached the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the 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.

### 41.2 WWDG main features

- Programmable free-running downcounter
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes lower than 0x40
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window (see [Figure 419](#))
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the downcounter is equal to 0x40.

### 41.3 WWDG functional description

If the watchdog is activated (the WDGA bit is set in the WWDG\_CR register) and when the 7-bit downcounter (T[6:0] bits) 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.

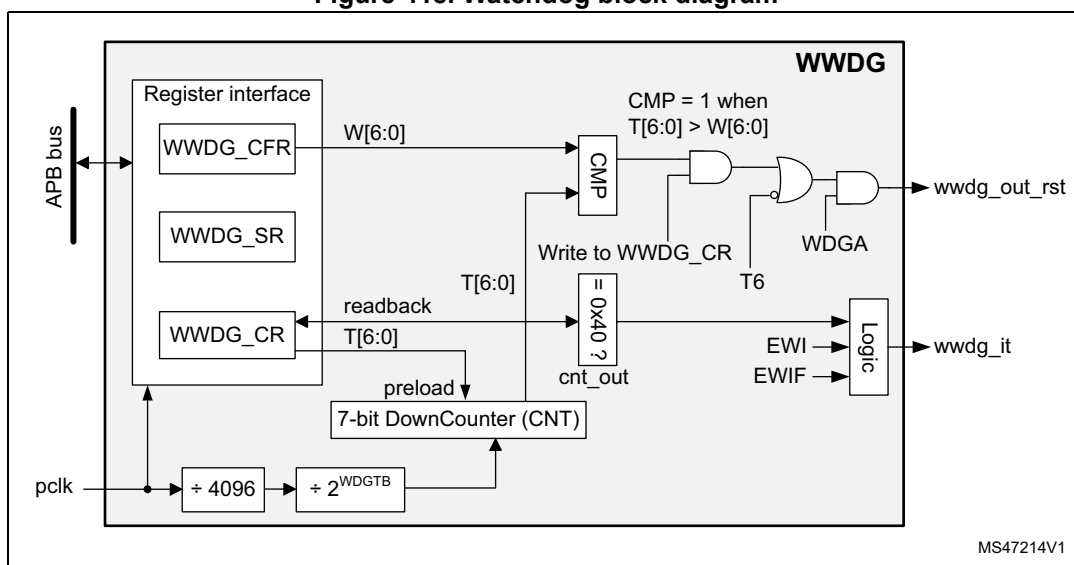
The application program must write in the WWDG\_CR register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the counter value is lower than the window register value and higher than 0x3F. The value to be stored in the WWDG\_CR register must be between 0xFF and 0xC0.

Refer to [Figure 418](#) for the WWDG block diagram.



### 41.3.1 WWDG block diagram

Figure 418. Watchdog block diagram



### 41.3.2 Enabling the watchdog

The watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG\_CR register, then it cannot be disabled again except by a reset.

### 41.3.3 Controlling the downcounter

This downcounter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments which represents the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WWDG\_CR register (see [Figure 419](#)). The Configuration register (WWDG\_CFR) contains the high limit of the window: To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x3F. [Figure 419](#) 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).

### 41.3.4 Advanced watchdog interrupt feature

The Early Wakeup Interrupt (EWI) can be used if specific safety operations or data logging must be performed before the actual reset is generated. The EWI interrupt is enabled by setting the EWI bit in the WWDG\_CFR register. When the downcounter reaches the value 0x40, an EWI interrupt is generated and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case, the corresponding interrupt service routine (ISR) should reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDG\_SR register.

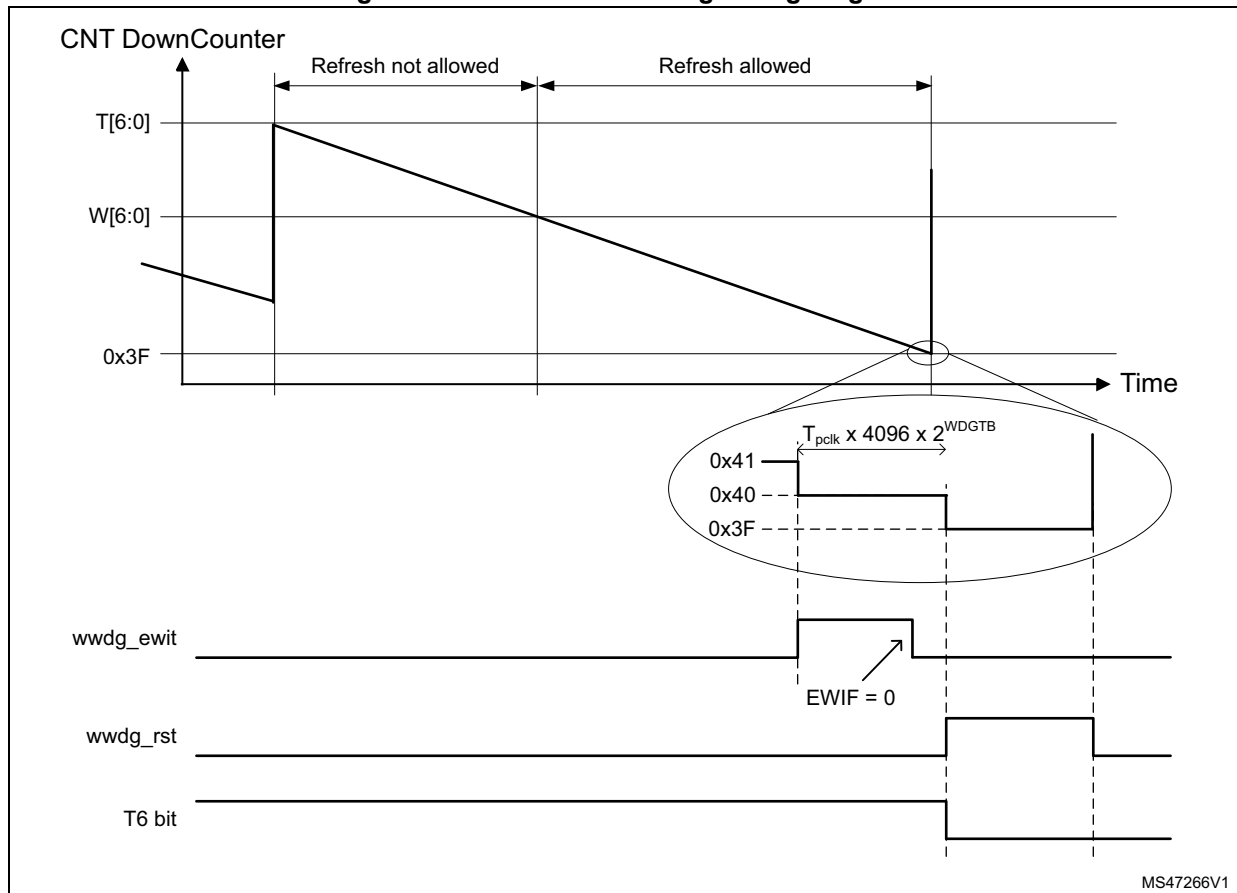
*Note:* When the EWI interrupt cannot be served, e.g. due to a system lock in a higher priority task, the WWDG reset will eventually be generated.

### 41.3.5 How to program the watchdog timeout

Use the formula in [Figure 419](#) 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 419. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[1:0]} \times (T[5:0] + 1) \quad (\text{ms})$$

where:

$t_{\text{WWDG}}$ : WWDG timeout

$t_{\text{PCLK}}$ : APB clock period measured in ms

4096: value corresponding to internal divider

As an example, let's assume APB frequency is equal to 48 MHz, WDG TB[1:0] is set to 3 and T[5:0] is set to 63:

$$t_{\text{WWDG}} = (1 / 48000) \times 4096 \times 2^3 \times (63 + 1) = 43.69 \text{ms}$$

Refer to the datasheet for the minimum and maximum values of the  $t_{\text{WWDG}}$ .

### 41.3.6 Debug mode

When the microcontroller 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 46.16.2: Debug support for timers, RTC, watchdog, bxCAN and I2C](#).

## 41.4 WWDG registers

Refer to [Section 1.2 on page 75](#) for a list of abbreviations used in register descriptions.  
 The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

### 41.4.1 Control register (WWDG\_CR)

Address offset: 0x000

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA		T[6:0]					
								rs	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **WDGA**: Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

- 0: Watchdog disabled
- 1: Watchdog enabled

Bits 6:0 **T[6:0]**: 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every  $(4096 \times 2^{WDGTB[1:0]})$  PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

### 41.4.2 Configuration register (WWDG\_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EWI	WDGTB[1:0]		W[6:0]						
						rs	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **EWI**: Early wakeup interrupt

When set, an interrupt occurs whenever the counter reaches the value 0x40. This interrupt is only cleared by hardware after a reset.

Bits 8:7 **WDGTB[1:0]**: Timer base

The time base of the prescaler can be modified as follows:

- 00: CK Counter Clock (PCLK div 4096) div 1
- 01: CK Counter Clock (PCLK div 4096) div 2
- 10: CK Counter Clock (PCLK div 4096) div 4
- 11: CK Counter Clock (PCLK div 4096) div 8

Bits 6:0 **W[6:0]**: 7-bit window value

These bits contain the window value to be compared with the downcounter.

### 41.4.3 Status register (WWDG\_SR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
															rc_w0

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EWIF**: Early wakeup interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing '0'. Writing '1' has no effect. This bit is also set if the interrupt is not enabled.

### 41.4.4 WWDG register map

The following table gives the WWDG register map and reset values.

**Table 241. 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	WWDG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x004	WWDG_CFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWI	WDGTB1	WDGTB0	W[6:0]						
	Reset value																							0	0	0	1	1	1	1	1	1	1
0x008	WWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
	Reset value																																0

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.

## 42 Real-time clock (RTC)

### 42.1 Introduction

The RTC provides an automatic wakeup to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

The RTC includes also a periodic programmable wakeup flag with interrupt capability.

Two 32-bit registers contain the seconds, minutes, hours (12- or 24-hour format), day (day of week), date (day of month), month, and year, expressed in binary coded decimal format (BCD). The sub-seconds value is also available in binary format.

Compensations for 28-, 29- (leap year), 30-, and 31-day months are performed automatically. Daylight saving time compensation can also be performed.

Additional 32-bit registers contain the programmable alarm subseconds, seconds, minutes, hours, day, and date.

A digital calibration feature is available to compensate for any deviation in crystal oscillator accuracy.

After Backup domain reset, all RTC registers are protected against possible parasitic write accesses.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

## 42.2 RTC main features

The RTC unit main features are the following (see [Figure 420: RTC block diagram](#)):

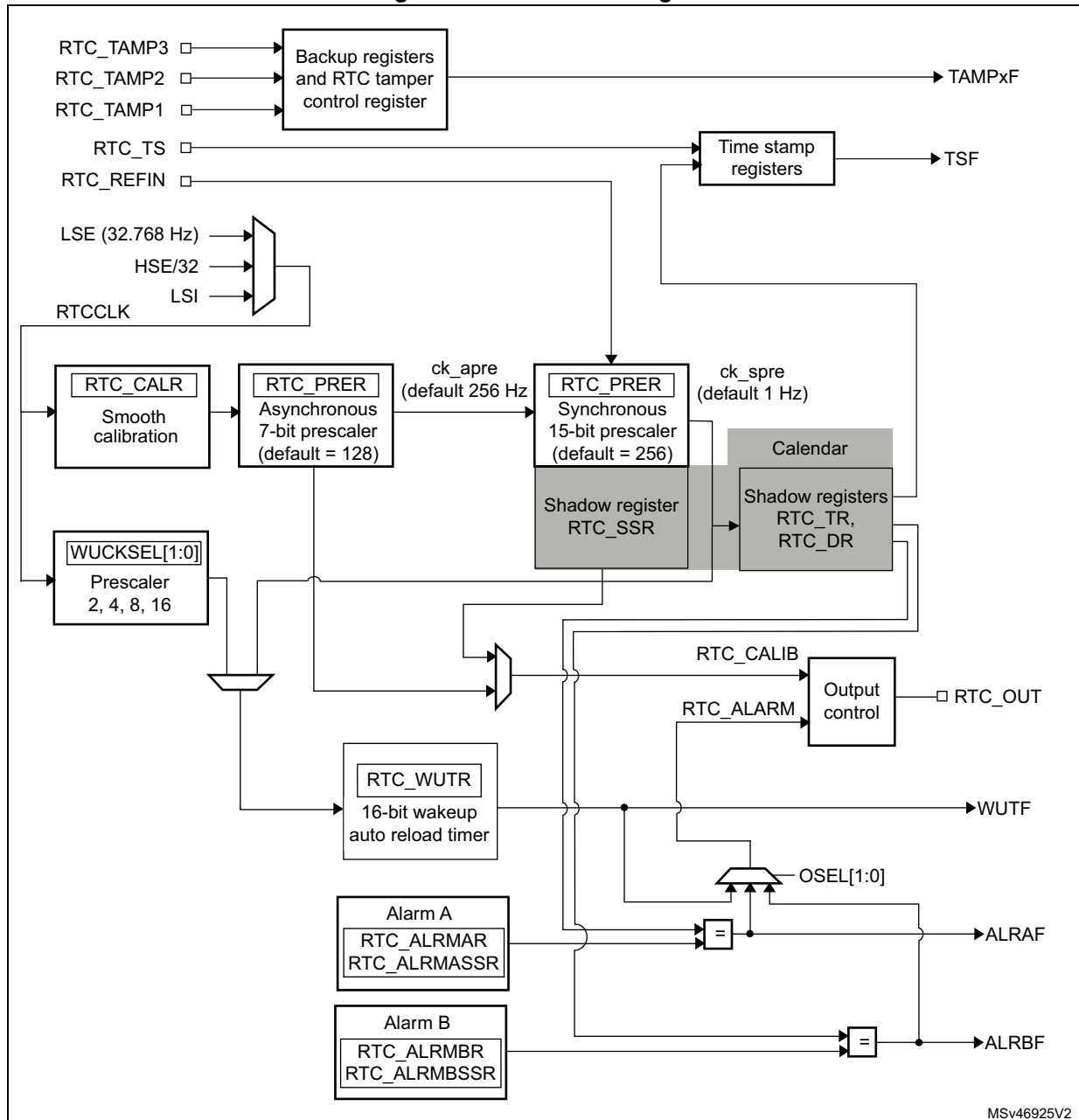
- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), day (day of week), date (day of month), month, and year.
- Daylight saving compensation programmable by software.
- Programmable alarm with interrupt function. The alarm can be triggered by any combination of the calendar fields.
- Automatic wakeup unit generating a periodic flag that triggers an automatic wakeup interrupt.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Accurate synchronization with an external clock using the subsecond shift feature.
- Digital calibration circuit (periodic counter correction): 0.95 ppm accuracy, obtained in a calibration window of several seconds
- Time-stamp function for event saving
- Tamper detection event with configurable filter and internal pull-up
- Maskable interrupts/events:
  - Alarm A
  - Alarm B
  - Wakeup interrupt
  - Time-stamp
  - Tamper detection
- 32 backup registers.



## 42.3 RTC functional description

### 42.3.1 RTC block diagram

Figure 420. RTC block diagram



MSv46925V2

The RTC includes:

- Two alarms
- Three tamper events from I/Os
  - 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
- 32 x 32-bit backup registers
  - The backup registers (RTC\_BKPxR) are implemented in the RTC domain that remains powered-on by VBAT when the VDD power is switched off.
- Output functions: RTC\_OUT which selects one of the following two outputs:
  - RTC\_CALIB: 512 Hz or 1Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC\_CR register.
  - RTC\_ALARM: This output is enabled by configuring the OSEL[1:0] bits in the RTC\_CR register which select the Alarm A, Alarm B or Wakeup outputs.
- Input functions:
  - RTC\_TS: timestamp event
  - RTC\_TAMP1: tamper1 event detection
  - RTC\_TAMP2: tamper2 event detection
  - RTC\_TAMP3: tamper3 event detection
  - RTC\_REFIN: 50 or 60 Hz reference clock input

### 42.3.2 GPIOs controlled by the RTC

RTC\_OUT, RTC\_TS and RTC\_TAMP1 are mapped on the same pin (PC13). PC13 pin configuration is controlled by the RTC, whatever the PC13 GPIO configuration, except for the RTC\_ALARM output open-drain mode. In this particular case, the GPIO must be configured as input. The RTC functions mapped on PC13 are available in all low-power modes and in VBAT mode.

The output mechanism follows the priority order shown in [Table 242](#).

**Table 242. RTC pin PC13 configuration<sup>(1)</sup>**

PC13 Pin configuration and function	OSEL[1:0] bits (RTC_ALARM output enable)	COE bit (RTC_CALIB output enable)	RTC_OUT _RMP bit	RTC_ALARM _TYPE bit	TAMP1E bit (RTC_TAMP1 input enable)	TSE bit (RTC_TS input enable)
RTC_ALARM output OD	01 or 10 or 11	Don't care	0	0	Don't care	Don't care
			1			
RTC_ALARM output PP	01 or 10 or 11	Don't care	0	1	Don't care	Don't care
			1			
RTC_CALIB output PP	00	1	0	Don't care	Don't care	Don't care

**Table 242. RTC pin PC13 configuration<sup>(1)</sup> (continued)**

PC13 Pin configuration and function	OSEL[1:0] bits (RTC_ALARM output enable)	COE bit (RTC_CALIB output enable)	RTC_OUT_RMP bit	RTC_ALARM_TYPE bit	TAMP1E bit (RTC_TAMP1 input enable)	TSE bit (RTC_TS input enable)
RTC_TAMP1 input floating	00	0	Don't care	Don't care	1	0
	00	1	1			
	01 or 10 or 11	0				
RTC_TS and RTC_TAMP1 input floating	00	0	Don't care	Don't care	1	1
	00	1	1			
	01 or 10 or 11	0				
RTC_TS input floating	00	0	Don't care	Don't care	0	1
	00	1	1			
	01 or 10 or 11	0				
Wakeup pin or Standard GPIO	00	0	Don't care	Don't care	0	0
	00	1	1			
	01 or 10 or 11	0				

1. OD: open drain; PP: push-pull.

In addition, it is possible to remap RTC\_OUT on PB2 pin thanks to RTC\_OUT\_RMP bit. In this case it is mandatory to configure PB2 GPIO registers as alternate function with the correct type. The remap functions are shown in [Table 243](#).

**Table 243. RTC\_OUT mapping**

OSEL[1:0] bits (RTC_ALARM output enable)	COE bit (RTC_CALIB output enable)	RTC_OUT_RMP bit	RTC_OUT on PC13	RTC_OUT on PB2
00	0	0	-	-
00	1		RTC_CALIB	-
01 or 10 or 11	Don't care		RTC_ALARM	-
00	0	1	-	-
00	1		-	RTC_CALIB
01 or 10 or 11	0		-	RTC_ALARM
01 or 10 or 11	1		RTC_ALARM	RTC_CALIB

The table below summarizes the RTC pins and functions capability in all modes.

Table 244. RTC functions over modes

Pin	RTC functions	Functional in all low-power modes except Standby/Shutdown modes	Functional in Standby/Shutdown mode	Functional in V <sub>BAT</sub> mode
PC13	RTC_TAMP1 RTC_TS RTC_OUT	YES	YES	YES
PA0	RTC_TAMP2	YES	YES	YES
PE6	RTC_TAMP3	YES	YES	YES
PB2	RTC_OUT	YES	NO	NO
PB15	RTC_REFIN	YES	NO	NO

### 42.3.3 Clock and prescalers

The RTC clock source (RTCCLK) is selected through the clock controller among the LSE clock, the LSI oscillator clock, and the HSE clock. For more information on the RTC clock source configuration, refer to [Section 6: 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 420: 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<sup>22</sup>.

This corresponds to a maximum input frequency of around 4 MHz.

f<sub>ck\_apre</sub> is given by the following formula:

$$f_{CK\_APRE} = \frac{f_{RTCCLK}}{PREDIV\_A + 1}$$

The ck\_apre clock is used to clock the binary RTC\_SSR subseconds downcounter. When it reaches 0, RTC\_SSR is reloaded with the content of PREDIV\_S.

f<sub>ck\_spre</sub> is given by the following formula:

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(PREDIV\_S + 1) \times (PREDIV\_A + 1)}$$

The `ck_spre` clock can be used either to update the calendar or as timebase for the 16-bit wakeup auto-reload timer. To obtain short timeout periods, the 16-bit wakeup auto-reload timer can also run with the `RTCCLK` divided by the programmable 4-bit asynchronous prescaler (see [Section 42.3.6: Periodic auto-wakeup](#) for details).

### 42.3.4 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with `PCLK` (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- `RTC_SSR` for the subseconds
- `RTC_TR` for the time
- `RTC_DR` for the date

Every two `RTCCLK` periods, the current calendar value is copied into the shadow registers, and the `RSF` bit of `RTC_ISR` register is set (see [Section 42.6.4: RTC initialization and status register \(RTC\\_ISR\)](#)). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 2 `RTCCLK` periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the `BYPHAD` control bit in the `RTC_CR` register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the `RTC_SSR`, `RTC_TR` or `RTC_DR` registers in `BYPHAD=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.

### 42.3.5 Programmable alarms

The RTC unit provides programmable alarm: Alarm A and Alarm B. The description below is given for Alarm A, but can be translated in the same way for Alarm B.

The programmable alarm function is enabled through the `ALRAE` bit in the `RTC_CR` register. The `ALRAF` is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers `RTC_ALRMASR` and `RTC_ALRMAR`. Each calendar field can be independently selected through the `MSKx` bits of the `RTC_ALRMAR` register, and through the `MASKSSx` bits of the `RTC_ALRMASR` register. The alarm interrupt is enabled through the `ALRAIE` bit in the `RTC_CR` register.

**Caution:** If the seconds field is selected (`MSK1` bit reset in `RTC_ALRMAR`), the synchronous prescaler division factor set in the `RTC_PRER` register must be at least 3 to ensure correct behavior.

Alarm A and Alarm B (if enabled by bits `OSEL[1:0]` in `RTC_CR` register) can be routed to the `RTC_ALARM` output. `RTC_ALARM` output polarity can be configured through bit `POL` the `RTC_CR` register.

### 42.3.6 Periodic auto-wakeup

The periodic wakeup flag is generated by a 16-bit programmable auto-reload down-counter. The wakeup timer range can be extended to 17 bits.

The wakeup function is enabled through the `WUTE` bit in the `RTC_CR` register.

The wakeup timer clock input can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.  
When RTCCLK is LSE(32.768kHz), this allows to configure the wakeup interrupt period from 122  $\mu$ 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 wakeup time from 1 s to around 36 hours with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18h to 36h when WUCKSEL[2:1] = 11. In this last case 2<sup>16</sup> is added to the 16-bit counter current value. When the initialization sequence is complete (see [Programming the wakeup timer on page 1409](#)), the timer starts counting down. When the wakeup function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC\_ISR register, and the wakeup counter is automatically reloaded with its reload value (RTC\_WUTR register value).

The WUTF flag must then be cleared by software.

When the periodic wakeup interrupt is enabled by setting the WUTIE bit in the RTC\_CR register, it can exit the device from low-power modes.

The periodic wakeup flag can be routed to the RTC\_ALARM output provided it has been enabled through bits OSEL[1:0] of RTC\_CR register. RTC\_ALARM output polarity can be configured through the POL bit in the RTC\_CR register.

System reset, as well as low-power modes (Sleep, Stop and Standby) have no influence on the wakeup timer.

### 42.3.7 RTC initialization and configuration

#### RTC register access

The RTC registers are 32-bit registers. The APB interface introduces 2 wait-states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD=0.

#### RTC register write protection

After system reset, the RTC registers are protected against parasitic write access by clearing the DBP bit in the PWR\_CR1 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\_TAMPCR, 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\_ALRMASR/RTC\_ALRMAR).
3. Set ALRAE in the RTC\_CR register to enable Alarm A again.

*Note:* Each change of the RTC\_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

### Programming the wakeup timer

The following sequence is required to configure or change the wakeup timer auto-reload value (WUT[15:0] in RTC\_WUTR):

1. Clear WUTE in RTC\_CR to disable the wakeup timer.
2. Poll WUTWF until it is set in RTC\_ISR to make sure the access to wakeup auto-reload counter and to WUCKSEL[2:0] bits is allowed. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. Program the wakeup auto-reload value WUT[15:0], and the wakeup clock selection (WUCKSEL[2:0] bits in RTC\_CR). Set WUTE in RTC\_CR to enable the timer again. The wakeup timer restarts down-counting. The WUTWF bit is cleared up to 2 RTCCLK clock cycles after WUTE is cleared, due to clock synchronization.

### 42.3.8 Reading the calendar

#### When BYPSHAD control bit is cleared in the RTC\_CR register

To read the RTC calendar registers (RTC\_SSR, RTC\_TR and RTC\_DR) properly, the 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 two RTCCLK cycles. To ensure consistency between the 3 values, reading either RTC\_SSR or RTC\_TR locks the values in the higher-order calendar shadow registers until RTC\_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 2 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC\_SSR, RTC\_TR and RTC\_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers.

The RSF bit must be cleared after wakeup and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to [Calendar initialization and configuration on page 1409](#)): the software must wait until RSF is set before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers.

After synchronization (refer to [Section 42.3.10: RTC synchronization](#)): the software must wait until RSF is set before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers.

#### When the BYPSHAD control bit is set in the RTC\_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting



from low-power modes (STOP or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

*Note:* While `BYPSHAD=1`, instructions which read the calendar registers require one extra APB cycle to complete.

### 42.3.9 Resetting the RTC

The calendar shadow registers (RTC\_SSR, RTC\_TR and RTC\_DR) and some bits of the RTC status register (RTC\_ISR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a Backup domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC\_CR), the prescaler register (RTC\_PRER), the RTC calibration register (RTC\_CALR), the RTC shift register (RTC\_SHIFTR), the RTC timestamp registers (RTC\_TSSSR, RTC\_TSTR and RTC\_TSDR), the RTC tamper configuration register (RTC\_TAMPCR), the RTC backup registers (RTC\_BKPxR), the wakeup timer register (RTC\_WUTR), the Alarm A and Alarm B registers (RTC\_ALRMASR/RTC\_ALRMAR and RTC\_ALRMBSSR/RTC\_ALRMBR), 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.

### 42.3.10 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (RTC\_SSR or RTC\_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC\_SHIFTR.

RTC\_SSR contains the value of the synchronous prescaler counter. This allows one to calculate the exact time being maintained by the RTC down to a resolution of  $1 / (\text{PREDIV}_S + 1)$  seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV\_S[14:0]). The maximum resolution allowed (30.52  $\mu$ s with a 32768 Hz clock) is obtained with PREDIV\_S set to 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.

### 42.3.11 RTC reference clock detection

The update of the RTC calendar can be synchronized to a reference clock, RTC\_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC\_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC\_REFIN detection is enabled (REFCKON bit of RTC\_CR set to 1), the calendar is still clocked by the LSE, and RTC\_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC\_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck\_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck\_apre periods when detecting the first reference clock edge. A smaller window of 3 ck\_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the 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.*

### 42.3.12 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual RTCCLK pulses). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

The smooth digital calibration is performed during a cycle of about  $2^{20}$  RTCCLK pulses, or 32 seconds when the input frequency is 32768 Hz. This cycle is maintained by a 20-bit counter, `cal_cnt[19:0]`, clocked by RTCCLK.

The smooth calibration register (RTC\_CALR) specifies the number of RTCCLK clock cycles to be masked during the 32-second cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the 32-second cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

*Note:* CALM[8:0] (RTC\_CALR) specifies the number of RTCCLK pulses to be masked during the 32-second cycle. Setting the bit CALM[0] to '1' causes exactly one pulse to be masked during the 32-second cycle at the moment when `cal_cnt[19:0]` is 0x80000; CALM[1]=1 causes two other cycles to be masked (when `cal_cnt` is 0x40000 and 0xC0000); CALM[2]=1 causes four other cycles to be masked (`cal_cnt` = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8]=1 which causes 256 clocks to be masked (`cal_cnt` = 0xXX800).

While CALM allows the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to '1' effectively inserts an extra RTCCLK pulse every  $2^{11}$  RTCCLK cycles, which means that 512 clocks are added during every 32-second cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 RTCCLK cycles can be added during the 32-second cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (FCAL) given the input frequency (FRTCCLK) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

#### Calibration when PREDIV\_A < 3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV\_A bits in RTC\_PRER register) is less than 3. If CALP was already set to 1 and PREDIV\_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

To perform a calibration with PREDIV\_A less than 3, the synchronous prescaler value (PREDIV\_S) should be reduced so that each second is accelerated by 8 RTCCLK clock cycles, which is equivalent to adding 256 clock cycles every 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.

### 42.3.13 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 42.6.16: RTC tamper configuration register \(RTC\\_TAMPCR\)](#).

### 42.3.14 Tamper detection

The RTC\_TAMPx input events can be configured either for edge detection, or for level detection with filtering.

The tamper detection can be configured for the following purposes:

- erase the RTC backup registers (default configuration)
- generate an interrupt, capable to wakeup 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 42.6.20: RTC backup registers \(RTC\\_BKPxR\)](#) and [Tamper detection initialization on page 1416](#))

except if the TAMPxNOERASE bit is set, or if TAMPxMF is set in the RTC\_TAMPCR register.

### Tamper detection initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the RTC\_TAMPCR 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\_TAMPCR 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\_TAMPCR 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\_TAMPCR 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  $V_{DD}$  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.*

### 42.3.15 Calibration clock output

When the COE bit is set to 1 in the RTC\_CR register, a reference clock is provided on the RTC\_CALIB device output.

If the COSEL bit in the RTC\_CR register is reset and PREDIV\_A = 0x7F, the RTC\_CALIB frequency is  $f_{\text{RTCCLK}}/64$ . This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The RTC\_CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and "PREDIV\_S+1" is a non-zero multiple of 256 (i.e: PREDIV\_S[7:0] = 0xFF), the RTC\_CALIB frequency is  $f_{\text{RTCCLK}}/(256 * (\text{PREDIV\_A}+1))$ . This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV\_A = 0x7F, PREDIV\_S = 0xFF), with an RTCCLK frequency at 32.768 kHz. The 1 Hz output is affected when a shift operation is on going and may toggle during the shift operation (SHPF=1).

*Note:* When the `RTC_CALIB` or `RTC_ALARM` output is selected, the `RTC_OUT` pin is automatically configured as output.

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.

### 42.3.16 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 the control bit `RTC_ALARM_TYPE` in the `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).

When the `RTC_CALIB` or `RTC_ALARM` output is selected, the `RTC_OUT` pin is automatically configured as output.

## 42.4 RTC low-power modes

**Table 245. 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	Peripheral registers content is kept.
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.



## 42.5 RTC interrupts

All RTC interrupts are connected to the EXTI controller. Refer to [Section 13: Extended interrupts and events controller \(EXTI\)](#).

To enable the RTC Alarm interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the RTC Alarm event in interrupt mode and select the rising edge sensitivity.
2. Configure and enable the RTC\_ALARM IRQ channel in the NVIC.
3. Configure the RTC to generate RTC alarms.

To enable the RTC Tamper interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the RTC Tamper event in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the RTC\_TAMP\_STAMP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC tamper event.

To enable the RTC TimeStamp interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the RTC TimeStamp event in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the RTC\_TAMP\_STAMP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC time-stamp event.

To enable the Wakeup timer interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the Wakeup timer even in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the RTC\_WKUP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC Wakeup timer event.

**Table 246. 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.

## 42.6 RTC registers

Refer to [Section 1.2 on page 75](#) of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

### 42.6.1 RTC time register (RTC\_TR)

The RTC\_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to [Calendar initialization and configuration on page 1409](#) and [Reading the calendar on page 1410](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 1408](#).

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31-23 Reserved, must be kept at reset value

Bit 22 **PM**: AM/PM notation

0: AM or 24-hour format  
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

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

### 42.6.2 RTC date register (RTC\_DR)

The RTC\_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to [Calendar initialization and configuration on page 1409](#) and [Reading the calendar on page 1410](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 1408](#).

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset: 0x0000 2101 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units

000: forbidden

001: Monday

...

111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

### 42.6.3 RTC control register (RTC\_CR)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
							r/w	r/w	r/w	r/w	r/w	r/w	r/w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYPS HAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w		r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:25 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: Wakeup output enabled

Bit 20 **POL**: Output polarity

This bit is used to configure the polarity of RTC\_ALARM output

- 0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0])
- 1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]).

Bit 19 **COSEL**: Calibration output selection

When COE=1, this bit selects which signal is output on RTC\_CALIB.

- 0: Calibration output is 512 Hz (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 42.3.15: Calibration clock output](#)

Bit 18 **BKP**: Backup

This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

- Bit 17 **SUB1H**: Subtract 1 hour (winter time change)  
When this bit is set, 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**: Wakeup timer interrupt enable  
0: Wakeup timer interrupt disabled  
1: Wakeup timer interrupt enabled
- Bit 13 **ALRBIE**: Alarm B interrupt enable  
0: Alarm B Interrupt disable  
1: Alarm B Interrupt enable
- Bit 12 **ALRAIE**: Alarm A interrupt enable  
0: Alarm A interrupt disabled  
1: Alarm A interrupt enabled
- Bit 11 **TSE**: timestamp enable  
0: timestamp disable  
1: timestamp enable
- Bit 10 **WUTE**: Wakeup timer enable  
0: Wakeup timer disabled  
1: Wakeup timer enabled  
*Note: When the wakeup 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 **BYPHAD**: Bypass the shadow registers

0: Calendar values (when reading from RTC\_SSR, RTC\_TR, and RTC\_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.

1: Calendar values (when reading from RTC\_SSR, RTC\_TR, and RTC\_DR) are taken directly from the calendar counters.

*Note: If the frequency of the 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]**: Wakeup clock selection

000: RTC/16 clock is selected

001: RTC/8 clock is selected

010: RTC/4 clock is selected

011: RTC/2 clock is selected

10x: ck\_spre (usually 1 Hz) clock is selected

11x: ck\_spre (usually 1 Hz) clock is selected and  $2^{16}$  is added to the WUT counter value (see note below)

*Note: Bits 7, 6 and 4 of this register can be written in initialization mode only (RTC\_ISR/INITF = 1).*

*WUT = Wakeup unit counter value.  $WUT = (0x0000 \text{ to } 0xFFFF) + 0x10000$  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 1408](#).*

**Caution:** TSE must be reset when TSEDGE is changed to avoid spuriously setting of TSF.

### 42.6.4 RTC initialization and status register (RTC\_ISR)

This register is write protected (except for RTC\_ISR[13:8] bits). The write access procedure is described in [RTC register write protection on page 1408](#).

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**: Wakeup timer flag  
This flag is set by hardware when the wakeup auto-reload counter reaches 0.  
This flag is cleared by software by writing 0.  
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.
- Bit 9 **ALRBF**: Alarm B flag  
This flag is set by hardware when the time/date registers (RTC\_TR and RTC\_DR) match the Alarm B register (RTC\_ALRMBR).  
This flag is cleared by software by writing 0.
- Bit 8 **ALRAF**: Alarm A flag  
This flag is set by hardware when the time/date registers (RTC\_TR and RTC\_DR) match the Alarm A register (RTC\_ALRMAR).  
This flag is cleared by software by writing 0.
- Bit 7 **INIT**: Initialization mode  
0: Free running mode  
1: Initialization mode used to program time and date register (RTC\_TR and RTC\_DR), and prescaler register (RTC\_PRER). Counters are stopped and start counting from the new value when INIT is reset.
- Bit 6 **INITF**: Initialization flag  
When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.  
0: Calendar registers update is not allowed  
1: Calendar registers update is allowed
- Bit 5 **RSF**: Registers synchronization flag  
This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC\_SSRx, RTC\_TRx and RTC\_DRx). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF=1), or when in bypass shadow register mode (BYPSHAD=1). This bit can also be cleared by software.  
It is cleared either by software or by hardware in initialization mode.  
0: Calendar shadow registers not yet synchronized  
1: Calendar shadow registers synchronized
- Bit 4 **INITS**: Initialization status flag  
This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).  
0: Calendar has not been initialized  
1: Calendar has been initialized
- Bit 3 **SHPF**: Shift operation pending  
0: No shift operation is pending  
1: A shift operation is pending  
This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC\_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.



**Bit 2 WUTWF:** Wakeup timer write flag

This bit is set by hardware 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 wakeup timer values can be changed when WUTE bit is cleared and WUTWF is set.

0: Wakeup timer configuration update not allowed

1: Wakeup timer configuration update allowed

**Bit 1 ALRBWF:** Alarm B write flag

This bit is set by hardware when Alarm B values can be changed, after the ALRBE bit has been set to 0 in RTC\_CR.

It is cleared by hardware in initialization mode.

0: Alarm B update not allowed

1: Alarm B update allowed

**Bit 0 ALRAWF:** Alarm A write flag

This bit is set by hardware when Alarm A values can be changed, after the ALRAE bit has been set to 0 in RTC\_CR.

It is cleared by hardware in initialization mode.

0: Alarm A update not allowed

1: Alarm A update allowed

*Note:* The bits ALRAF, ALRBF, WUTF and TSF are cleared 2 APB clock cycles after programming them to 0.

### 42.6.5 RTC prescaler register (RTC\_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to [Calendar initialization and configuration on page 1409](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 1408](#).

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:

$$ck\_spre\ frequency = ck\_apre\ frequency / (PREDIV\_S + 1)$$

### 42.6.6 RTC wakeup timer register (RTC\_WUTR)

This register can be written only when WUTWF is set to 1 in RTC\_ISR.

This register is write protected. The write access procedure is described in [RTC register write protection on page 1408](#).

Address offset: 0x14

Backup domain reset value: 0x0000 FFFF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value

Bits 15:0 **WUT[15:0]**: Wakeup auto-reload value bits

When the wakeup timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck\_wut cycles. The ck\_wut period is selected through WUCKSEL[2:0] bits of the RTC\_CR register

When WUCKSEL[2] = 1, the wakeup timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs (WUT+1) ck\_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

### 42.6.7 RTC alarm A register (RTC\_ALRMAR)

This register can be written only when ALRAWF is set to 1 in RTC\_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in [RTC register write protection on page 1408](#).

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.

### 42.6.8 RTC alarm B register (RTC\_ALRMBR)

This register can be written only when ALRBWF is set to 1 in RTC\_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in [RTC register write protection on page 1408](#).

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

### 42.6.9 RTC write protection register (RTC\_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **KEY**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to [RTC register write protection](#) for a description of how to unlock RTC register write protection.

### 42.6.10 RTC sub second register (RTC\_SSR)

Address offset: 0x28

Backup domain reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits31:16 Reserved, must be kept at reset value

Bits 15:0 **SS**: Sub second value

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

$$\text{Second fraction} = (\text{PREDIV}_S - \text{SS}) / (\text{PREDIV}_S + 1)$$

*Note: SS can be larger than PREDIV\_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC\_TR/RTC\_DR.*

### 42.6.11 RTC shift control register (RTC\_SHIFTR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 1408](#).

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC\_ISR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value

Bits 14:0 **SUBFS**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC\_ISR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

$$\text{Delay (seconds)} = \text{SUBFS} / (\text{PREDIV\_S} + 1)$$

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

$$\text{Advance (seconds)} = (1 - (\text{SUBFS} / (\text{PREDIV\_S} + 1)))$$

*Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF=1 to be sure that the shadow registers have been updated with the shifted time.*

### 42.6.12 RTC timestamp time register (RTC\_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC\_ISR. It is cleared when TSF bit is reset.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value

Bit 22 **PM**: AM/PM notation

0: AM or 24-hour format

1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.



### 42.6.13 RTC timestamp date register (RTC\_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC\_ISR. It is cleared when TSF bit is reset.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[1:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value

Bits 15:13 **WDU[1:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

**42.6.14 RTC time-stamp sub second register (RTC\_TSSSR)**

The content of this register is valid only when RTC\_ISR/TSF is set. It is cleared when the RTC\_ISR/TSF bit is reset.

Address offset: 0x38

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value

Bits 15:0 **SS**: Sub second value

SS[15:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

### 42.6.15 RTC calibration register (RTC\_CALR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 1408](#).

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:  $(512 * CALP) - CALM$ .

Refer to [Section 42.3.12: RTC smooth digital calibration](#).

Bit 14 **CALW8**: Use an 8-second calibration cycle period

When CALW8 is set to '1', the 8-second calibration cycle period is selected.

Note: CALM[1:0] are stuck at "00" when CALW8='1'. Refer to [Section 42.3.12: RTC smooth digital calibration](#).

Bit 13 **CALW16**: Use a 16-second calibration cycle period

When CALW16 is set to '1', the 16-second calibration cycle period is selected. This bit must not be set to '1' if CALW8=1.

Note: CALM[0] is stuck at '0' when CALW16='1'. Refer to [Section 42.3.12: RTC smooth digital calibration](#).

Bits 12:9 Reserved, must be kept at reset value

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of  $2^{20}$  RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See [Section 42.3.12: RTC smooth digital calibration on page 1413](#).

### 42.6.16 RTC tamper configuration register (RTC\_TAMPCR)

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	TAMPPRCH [1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMP TS	TAMP3 TRG	TAMP3 E	TAMP2 TRG	TAMP2 E	TAMPI 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 TAMPIE = 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 TAMPIE = 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 TAMPIE = 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 **TAMPTS**: Activate timestamp on tamper detection event  
 0: Tamper detection event does not cause a timestamp to be saved  
 1: Save timestamp on tamper detection event  
 TAMPTS is valid even if TSE=0 in the RTC\_CR register.
- Bit 6 **TAMP3TRG**: Active level for RTC\_TAMP3 input  
 if TAMPFLT ≠ 00:  
 0: RTC\_TAMP3 input staying low triggers a tamper detection event.  
 1: RTC\_TAMP3 input staying high triggers a tamper detection event.  
 if TAMPFLT = 00:  
 0: RTC\_TAMP3 input rising edge triggers a tamper detection event.  
 1: RTC\_TAMP3 input falling edge triggers a tamper detection event.

Bit 5 **TAMP3E**: RTC\_TAMP3 detection enable

0: RTC\_TAMP3 input detection disabled

1: RTC\_TAMP3 input detection enabled

Bit 4 **TAMP2TRG**: Active level for RTC\_TAMP2 input

if TAMPFLT != 00:

0: RTC\_TAMP2 input staying low triggers a tamper detection event.

1: RTC\_TAMP2 input staying high triggers a tamper detection event.

if TAMPFLT = 00:

0: RTC\_TAMP2 input rising edge triggers a tamper detection event.

1: RTC\_TAMP2 input falling edge triggers a tamper detection event.

Bit 3 **TAMP2E**: RTC\_TAMP2 input detection enable

0: RTC\_TAMP2 detection disabled

1: RTC\_TAMP2 detection enabled

Bit 2 **TAMPIE**: Tamper interrupt enable

0: Tamper interrupt disabled

1: Tamper interrupt enabled.

*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, TAMPxE must be reset when TAMPxTRG is changed to avoid spuriously setting TAMPxF.

### 42.6.17 RTC alarm A sub second register (RTC\_ALRMASSR)

This register can be written only when ALRAE is reset in RTC\_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in [RTC register write protection on page 1408](#)

Address offset: 0x44

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	Res.	Res.	Res.	MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.

Bits23:15 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.

### 42.6.18 RTC alarm B sub second register (RTC\_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC\_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).

Address offset: 0x48

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	Res.	Res.	Res.	MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	SS[14:0]															
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **MASKSS[3:0]**: Mask the most-significant bits starting at this bit

0x0: No comparison on sub seconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

0x1: SS[14:1] are don't care in Alarm B comparison. Only SS[0] is compared.

0x2: SS[14:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.

0x3: SS[14:3] are don't care in Alarm B comparison. Only SS[2:0] are compared.

...

0xC: SS[14:12] are don't care in Alarm B comparison. SS[11:0] are compared.

0xD: SS[14:13] are don't care in Alarm B comparison. SS[12:0] are compared.

0xE: SS[14] is don't care in Alarm B comparison. SS[13:0] are compared.

0xF: All 15 SS bits are compared and must match to activate alarm.

The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if Alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.



### 42.6.19 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTC_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':**

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 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

### 42.6.20 RTC backup registers (RTC\_BKPxR)

Address offset: 0x50 to 0xCC

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<sub>BAT</sub> when V<sub>DD</sub> 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 TAMPx F=1.



42.6.21 RTC register map

Table 247. RTC 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	RTC_TR	Res	Res	Res	Res	Res	Res	Res	Res	Res	PM	HT[1:0]	HU[3:0]			Res	MNT[2:0]			MNU[3:0]			Res	ST[2:0]			SU[3:0]													
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x04	RTC_DR	Res	Res	Res	Res	Res	Res	Res	Res	Res	YT[3:0]			YU[3:0]			WDU[2:0]		MT	MU[3:0]			Res	Res	DT[1:0]		DU[3:0]													
	Reset value										0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1		0	0	0	0	0	1							
0x08	RTC_CR	Res	Res	Res	Res	Res	Res	Res	Res	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	Res	FMT	BYPHAD	REFCKON	TSEGE	WUICKSEL[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						
0x0C	RTC_ISR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res							
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1						
0x10	RTC_PRER	Res	Res	Res	Res	Res	Res	Res	Res	Res	PREDIV_A[6:0]						PREDIV_S[14:0]																							
	Reset value										1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1					
0x14	RTC_WUTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	WUT[15:0]																					
	Reset value																		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1						
0x1C	RTC_ALRMAR	MSK4	WDSSEL	DT[1:0]		DU[3:0]			MSK3	PM	HT[1:0]	HU[3:0]			MSK2	MNT[2:0]		MNU[3:0]			MSK1	ST[2:0]		SU[3:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x20	RTC_ALRMBR	MSK4	WDSSEL	DT[1:0]		DU[3:0]			MSK3	PM	HT[1:0]	HU[3:0]			MSK2	MNT[2:0]		MNU[3:0]			MSK2	ST[2:0]		SU[3:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x24	RTC_WPR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	KEY													
	Reset value																										0	0	0	0	0	0	0	0						
0x28	RTC_SSR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SS[15:0]																					
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x2C	RTC_SHIFTR	ADD1S	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SUBFS[14:0]																					
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x30	RTC_TSTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	PM	HT[1:0]	HU[3:0]			Res	MNT[2:0]		MNU[3:0]			Res	ST[2:0]		SU[3:0]															
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								



Table 247. 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		
0x34	RTC_TSDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[1:0]	MT	MU[3:0]			Res.	Res.	DT[1:0]	DU[3:0]									
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x38	RTC_TSSSR	SS[15:0]																																	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x3C	RTC_CALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	Res.	Res.	Res.	Res.	CALM[8:0]										
	Reset value																	0	0	0					0	0	0	0	0	0	0	0	0		
0x40	RTC_TAMPCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3MF	TAMP3NOERASE	TAMP3IE	TAMP2MF	TAMP2NOERASE	TAMP2IE	TAMP1MF	TAMP1NOERASE	TAMP1IE	TAMPPUDIS	TAMPPRCH[1:0]	TAMPFLT[1:0]	TAMPFREQ[2:0]	TAMPTS	TAMP3TRG	TAMP3E	TAMP2TRG	TAMP2E	TAMP1TRG	TAMP1E	TAMP1TRG	TAMP1E					
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x44	RTC_ALRMASSR	Res.	Res.	Res.	Res.	MASKSS [3:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]																
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x48	RTC_ALRMBSSR	Res.	Res.	Res.	Res.	MASKSS [3:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]																
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
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.	Res.	Res.	RTC_OUT_RMP	RTC_ALARM_TYPE
	Reset value																																	0	0
0x50 to 0xCC	RTC_BKP0R	BKP[31:0]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50 to 0xCC	to RTC_BKP31R	BKP[31:0]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.



## 43 Inter-integrated circuit (I2C) interface

### 43.1 Introduction

The I<sup>2</sup>C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I<sup>2</sup>C bus. It provides multimaster capability, and controls all I<sup>2</sup>C bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

It is also SMBus (system management bus) and PMBus (power management bus) compatible.

DMA can be used to reduce CPU overload.

### 43.2 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 43.3: 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
- Wakeup from Stop mode on address match.

### 43.3 I2C implementation

This manual describes the full set of features implemented in I2C peripheral. In the STM32L4Rxxx and STM32L4Sxxx devices I2C1, I2C2, I2C3 and I2C4 implement the full set of features as shown in the following table, with the restriction that only I2C3 can wake up from Stop 2 mode.

**Table 248. I2C implementation**

I2C features <sup>(1)</sup>	I2C1	I2C2	I2C3	I2C4
7-bit addressing mode	X	X	X	X
10-bit addressing mode	X	X	X	X
Standard-mode (up to 100 kbit/s)	X	X	X	X
Fast-mode (up to 400 kbit/s)	X	X	X	X
Fast-mode Plus with 20mA output drive I/Os (up to 1 Mbit/s)	X	X	X	X
Independent clock	X	X	X	X
Wakeup from Stop 1 mode	X	X	X	X
Wakeup from Stop 2 mode	-	-	X	-
SMBus/PMBus	X	X	X	X

1. X = supported.

### 43.4 I2C functional description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the 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.

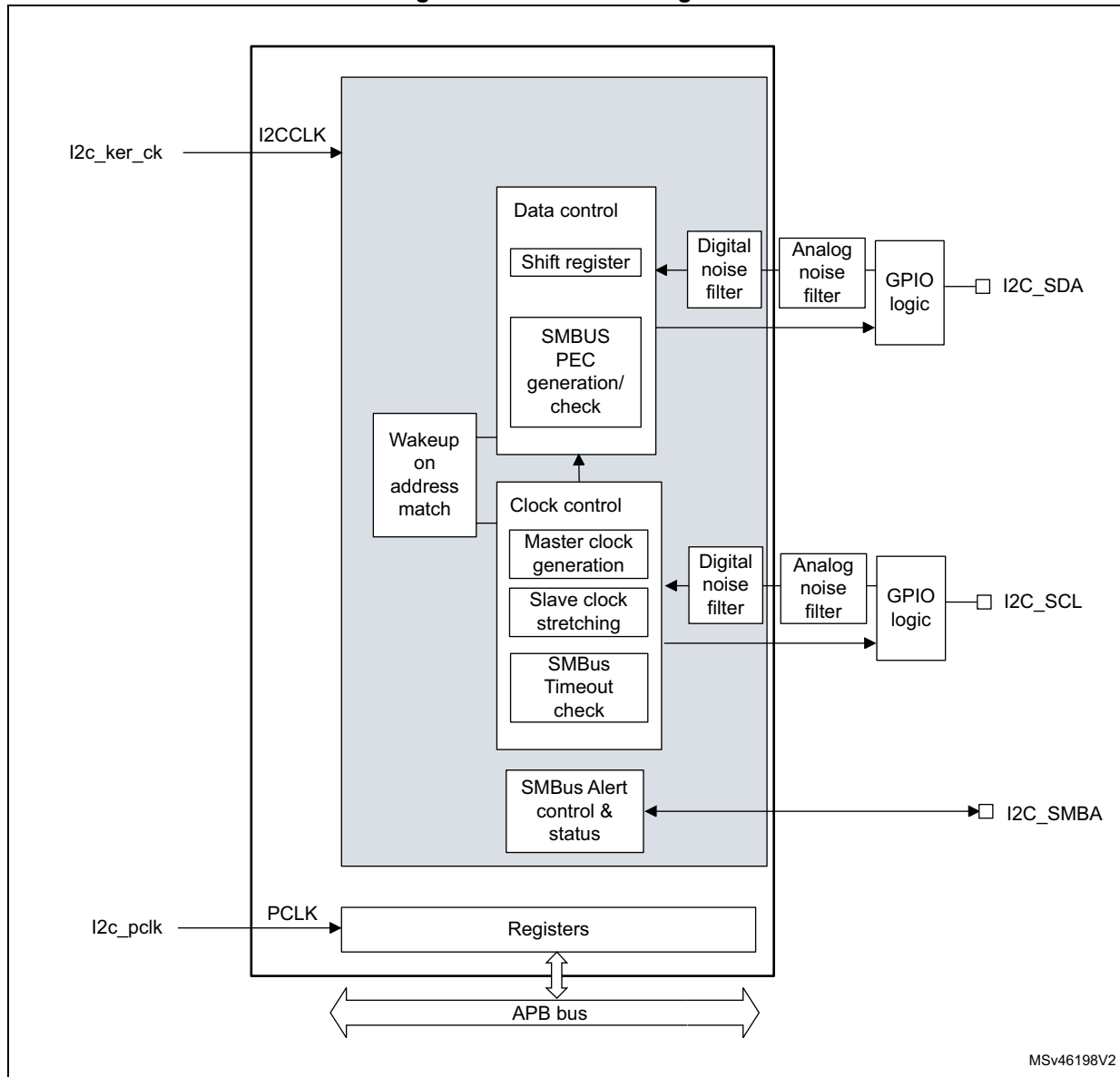
This interface can also be connected to a SMBus with the data pin (SDA) and clock pin (SCL).

If SMBus feature is supported: the additional optional SMBus Alert pin (SMBA) is also available.

### 43.4.1 I2C block diagram

The block diagram of the I2C interface is shown in [Figure 421](#).

**Figure 421. I2C block diagram**



The I2C is clocked by an independent clock source which allows to the I2C to operate independently from the PCLK frequency.

This independent clock source can be selected from the following three clock sources:

- PCLK1: APB1 clock (default value)
- HSI16: internal 16 MHz RC oscillator
- SYSCLK: system clock

Refer to [Section 6: Reset and clock control \(RCC\)](#) for more details.

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 43.3: I2C implementation](#).

### 43.4.2 I2C clock requirements

The I2C kernel is clocked by I2CCLK.

The I2CCLK period  $t_{I2CCLK}$  must respect the following conditions:

$$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4 \text{ and } t_{I2CCLK} < t_{HIGH}$$

with:

$t_{LOW}$ : SCL low time and  $t_{HIGH}$ : SCL high time

$t_{filters}$ : when enabled, sum of the delays brought by the analog filter and by the digital filter.

Analog filter delay is maximum 260 ns. Digital filter delay is  $DNF \times t_{I2CCLK}$ .

The PCLK clock period  $t_{PCLK}$  must respect the following condition:

$$t_{PCLK} < 4/3 t_{SCL}$$

with  $t_{SCL}$ : SCL period

**Caution:** When the I2C kernel is clocked by PCLK, this clock must respect the conditions for  $t_{I2CCLK}$ .

### 43.4.3 Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master when it generates a START condition, and from master to slave if an arbitration loss or a STOP generation occurs, allowing multimaster capability.

#### Communication flow

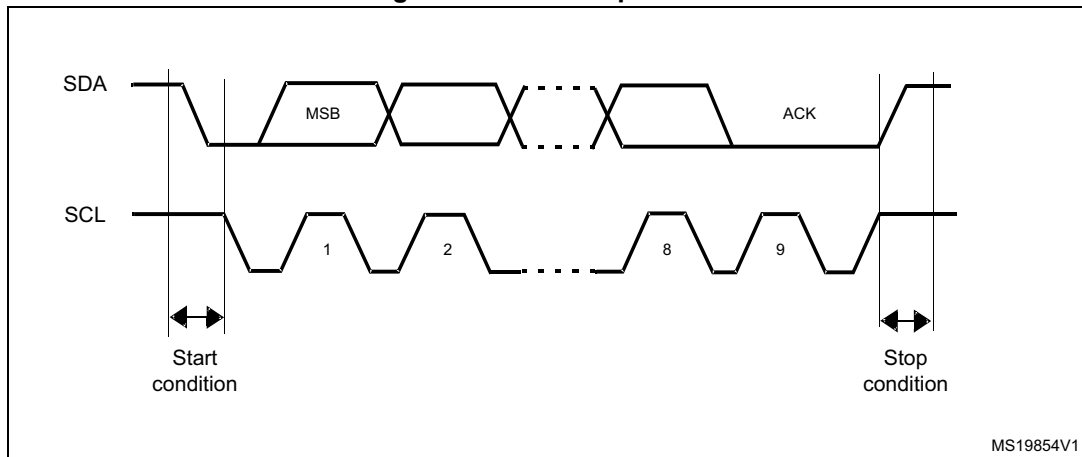
In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to the following figure.

Figure 422. I<sup>2</sup>C bus protocol



Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

#### 43.4.4 I2C initialization

##### Enabling and disabling the peripheral

The I2C peripheral clock must be configured and enabled in the clock controller (refer to [Section 6: Reset and clock control \(RCC\)](#)).

Then the I2C can be enabled by setting the PE bit in the I2C\_CR1 register.

When the I2C is disabled (PE=0), the I<sup>2</sup>C performs a software reset. Refer to [Section 43.4.5: Software reset](#) 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 to suppress spikes with a programmable length of 1 to 15 I2CCLK periods.



Table 249. 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 vs. standard requirements</li><li>– Stable length</li></ul>
Drawbacks	Variation vs. temperature, voltage, process	Wakeup from Stop mode on address match is not available when digital filter is enabled

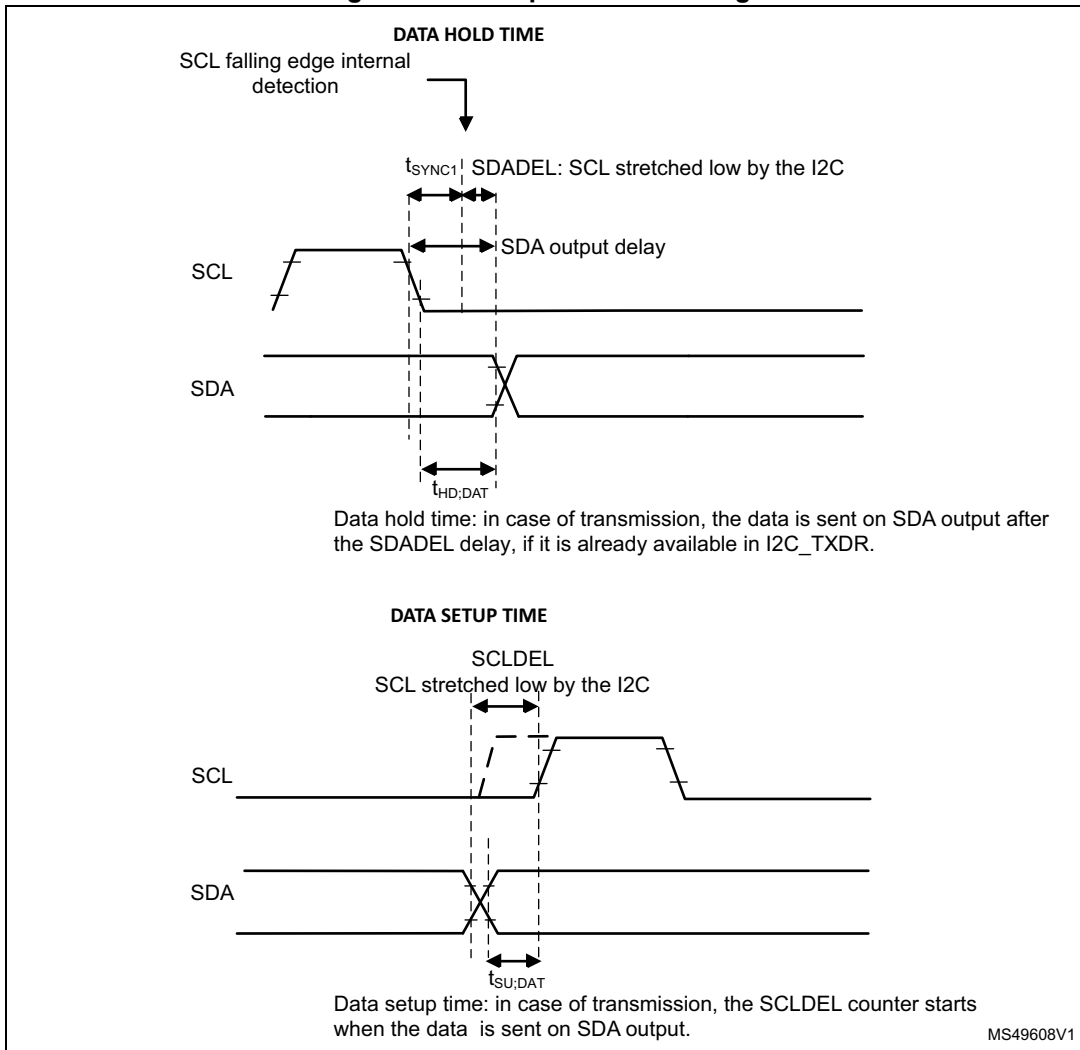
**Caution:** Changing the filter configuration is not allowed when the I2C is enabled.

### I2C timings

The timings must be configured in order to guarantee a correct data hold and setup time, used in master and slave modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C\_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C\_TIMINGR content in the I2C configuration window

**Figure 423. 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:  $t_{AF(min)} < t_{AF} < t_{AF(max)}$  ns.
- When enabled, input delay brought by the digital filter:  $t_{DNF} = DNF \times t_{I2CCLK}$
- Delay due to SCL synchronization to I2CCLK clock (2 to 3 I2CCLK periods)

In order to bridge the undefined region of the SCL falling edge, 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  $t_{HD;DAT}$  could 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 250: I2C-SMBUS specification data setup and hold times](#) 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 250: I2C-SMBUS specification data setup and hold times](#) 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, in case 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 250. 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	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	$\mu s$
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
$t_r$	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
$t_f$	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Additionally, in 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  $t_{HIGH}$ .

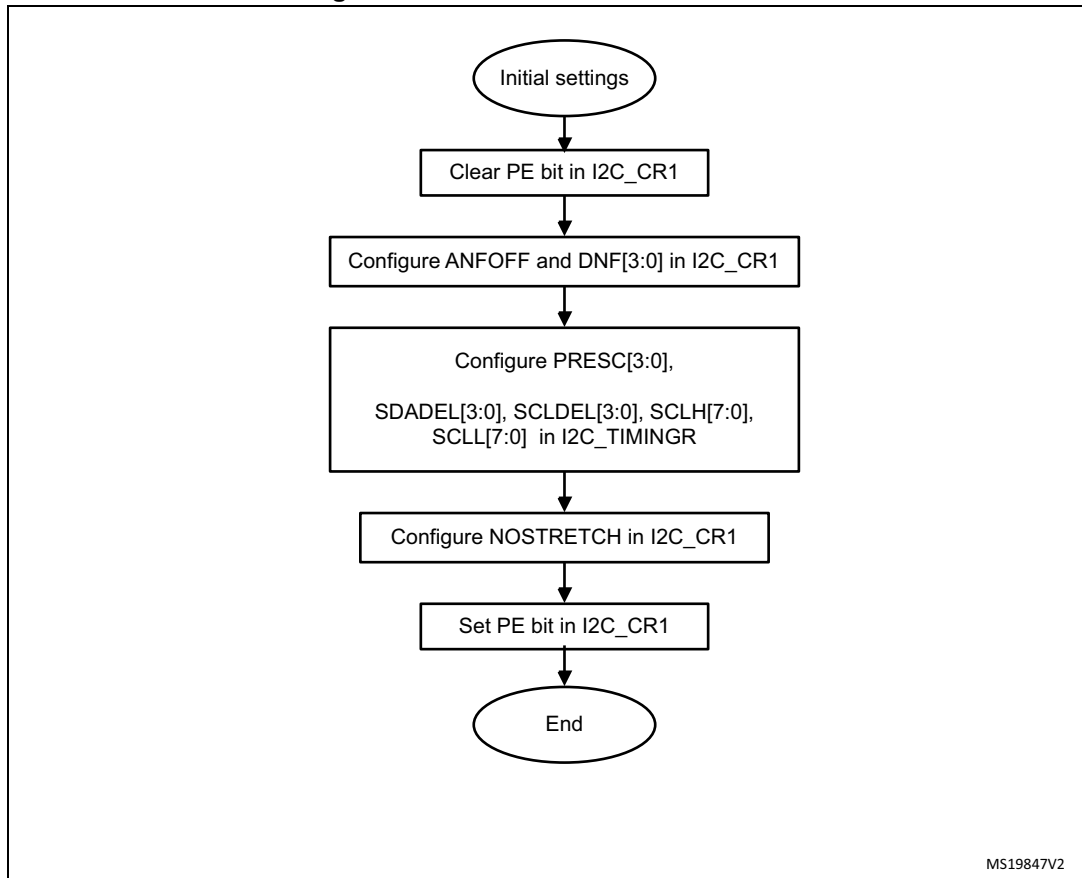
Refer to [I2C master initialization](#) for more details.

**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 424. I2C initialization flowchart



#### 43.4.5 Software reset

A software reset can be performed by clearing the PE bit in the I2C\_CR1 register. In that case I2C lines SCL and SDA are released. Internal states machines are reset and communication control bits, as well as status bits come back to their reset value. The configuration registers are not impacted.

Here is the list of impacted register bits:

1. I2C\_CR2 register: START, STOP, NACK
2. I2C\_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, OVR

and in addition when the SMBus feature is supported:

1. I2C\_CR2 register: PECBYTE
2. I2C\_ISR register: PECERR, TIMEOUT, ALERT

PE must be kept low during at least 3 APB clock cycles in order to perform the software reset. This is ensured by writing the following software sequence: - Write PE=0 - Check PE=0 - Write PE=1.

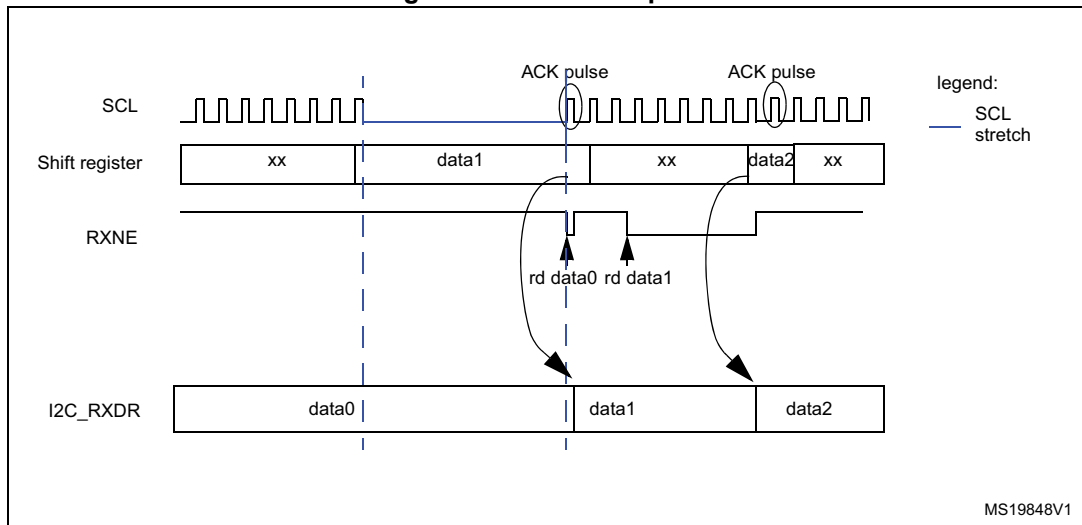
### 43.4.6 Data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

#### Reception

The SDA input fills the shift register. After the 8th SCL pulse (when the complete data byte is received), the shift register is copied into I2C\_RXDR register if it is empty (RXNE=0). If RXNE=1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C\_RXDR is read. The stretch is inserted between the 8th and 9th SCL pulse (before the Acknowledge pulse).

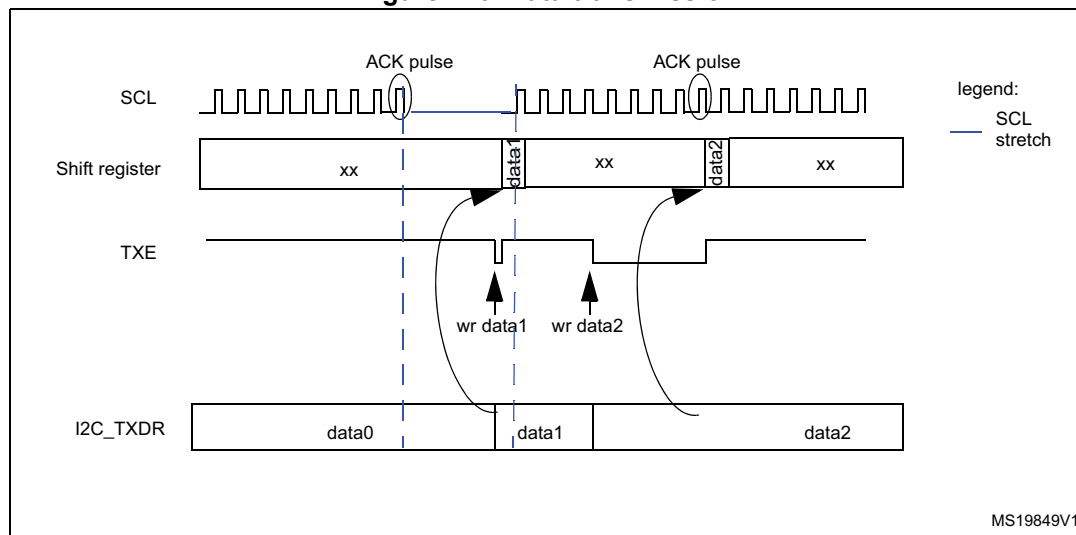
Figure 425. Data reception



## Transmission

If the I2C\_TXDR register is not empty (TXE=0), its content is copied into the shift register after the 9th SCL pulse (the Acknowledge pulse). Then the shift register content is shifted out on SDA line. If TXE=1, meaning that no data is written yet in I2C\_TXDR, SCL line is stretched low until I2C\_TXDR is written. The stretch is done after the 9th SCL pulse.

**Figure 426. Data transmission**



## Hardware transfer management

The I2C has a byte counter embedded in hardware in order to manage byte transfer and to close the communication in various modes such as:

- NACK, STOP and ReSTART generation in master mode
- ACK control in slave receiver mode
- PEC generation/checking when SMBus feature is supported

The byte counter is always used in master mode. By default it is disabled in slave mode, but it can be enabled by software by setting the SBC (Slave Byte Control) bit in the I2C\_CR2 register.

The number of bytes to be transferred is programmed in the NBYTES[7:0] bit field in the I2C\_CR2 register. If the number of bytes to be transferred (NBYTES) is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected by setting the RELOAD bit in the I2C\_CR2 register. In this mode, TCR flag is set when the number of bytes programmed in NBYTES has been transferred, and an interrupt is generated if TCIE is set. SCL is stretched as long as TCR flag is set. TCR is cleared by software when NBYTES is written to a non-zero value.

When the NBYTES counter is reloaded with the last number of bytes, RELOAD bit must be cleared.

When RELOAD=0 in master mode, the counter can be used in 2 modes:

- **Automatic end mode** (AUTOEND = '1' in the I2C\_CR2 register). In this mode, the master automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred.
- **Software end mode** (AUTOEND = '0' in the I2C\_CR2 register). In this mode, software action is expected once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit is set in the I2C\_CR2 register. This mode must be used when the master wants to send a RESTART condition.

**Caution:** The AUTOEND bit has no effect when the RELOAD bit is set.

**Table 251. 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

### 43.4.7 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 2nd 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 9th SCL pulse (ACK pulse) of the next data byte occurs. If not an overrun occurs, the OVR flag is set in the I2C\_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

### Slave Byte Control mode

In order to allow byte ACK control in slave reception mode, Slave Byte Control mode must be enabled by setting the SBC bit in the I2C\_CR1 register. This is required to be compliant with SMBus standards.

Reload mode must be selected in order to allow byte ACK control in slave reception mode (RELOAD=1). To get control of each byte, NBYTES must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the 8th and 9th SCL pulses. 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.

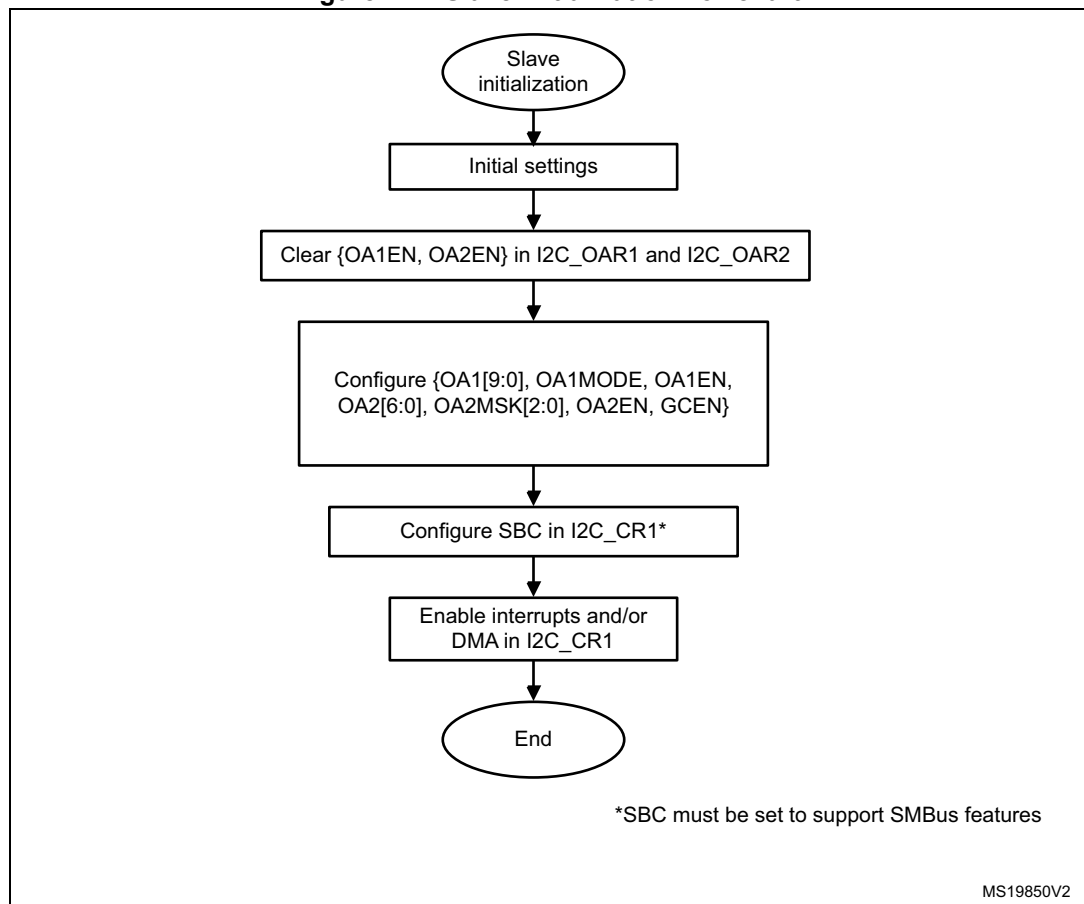
NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

**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:** Slave Byte Control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH=1 is not allowed.

**Figure 427. Slave initialization flowchart**



### Slave transmitter

A transmit interrupt status (TXIS) is generated when the I2C\_TXDR register becomes empty. An interrupt is generated if the TXIE bit is set in the I2C\_CR1 register.

The TXIS bit is cleared when the I2C\_TXDR register is written with the next data byte to be transmitted.

When a NACK is received, the NACKF bit is set in the I2C\_ISR register and an interrupt is generated if the NACKIE bit is set in the I2C\_CR1 register. The slave automatically releases the SCL and SDA lines in order to let the master perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When a STOP is received and the STOPIE bit is set in the I2C\_CR1 register, the STOPF flag is set in the I2C\_ISR register and an interrupt is generated. In most applications, the SBC bit is usually programmed to '0'. In this case, if TXE = 0 when the slave address is received (ADDR=1), 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 will be 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 428. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=0

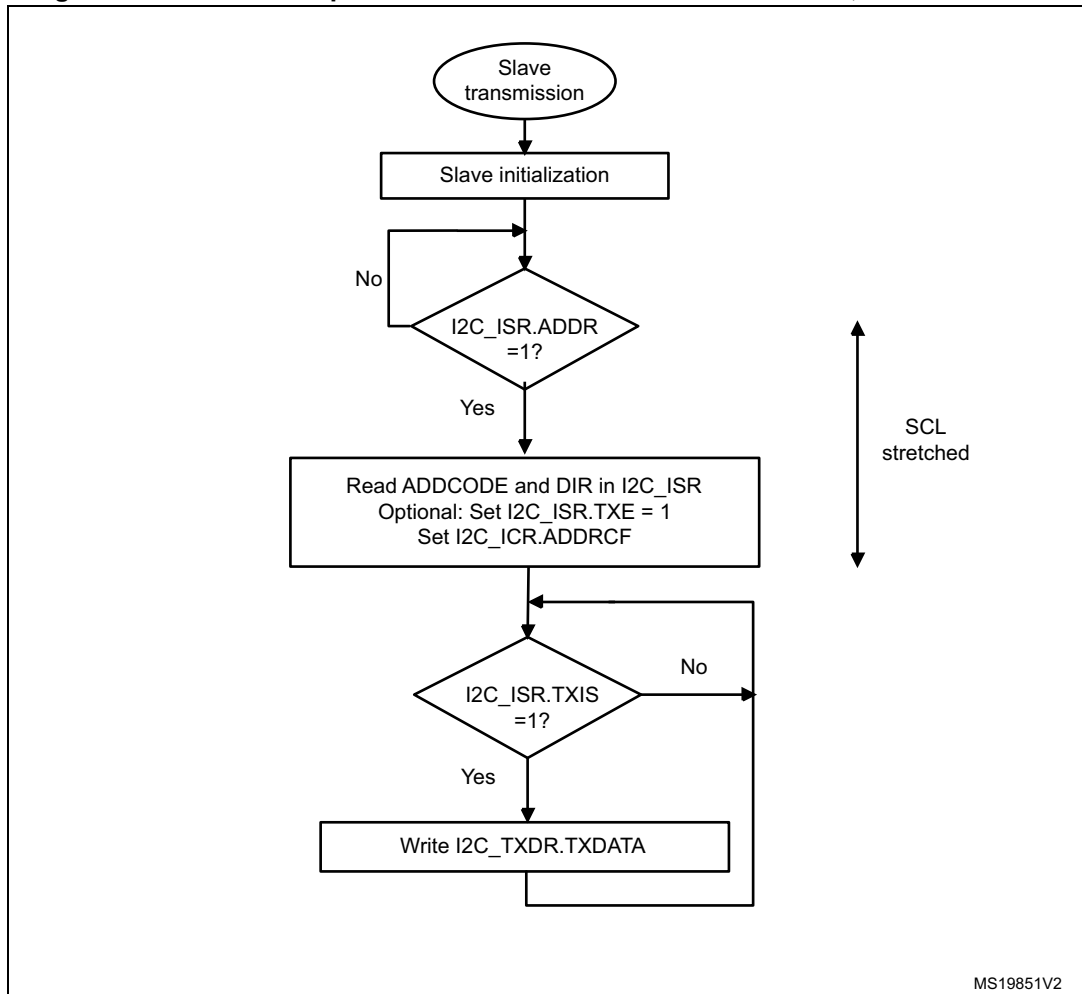


Figure 429. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=1

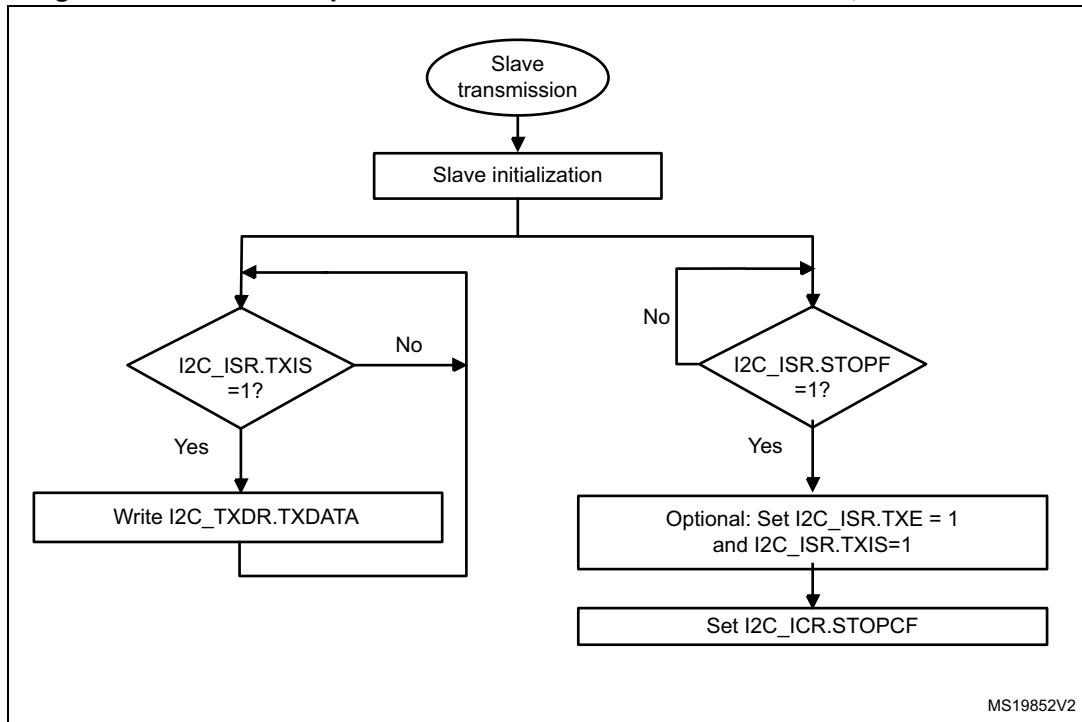
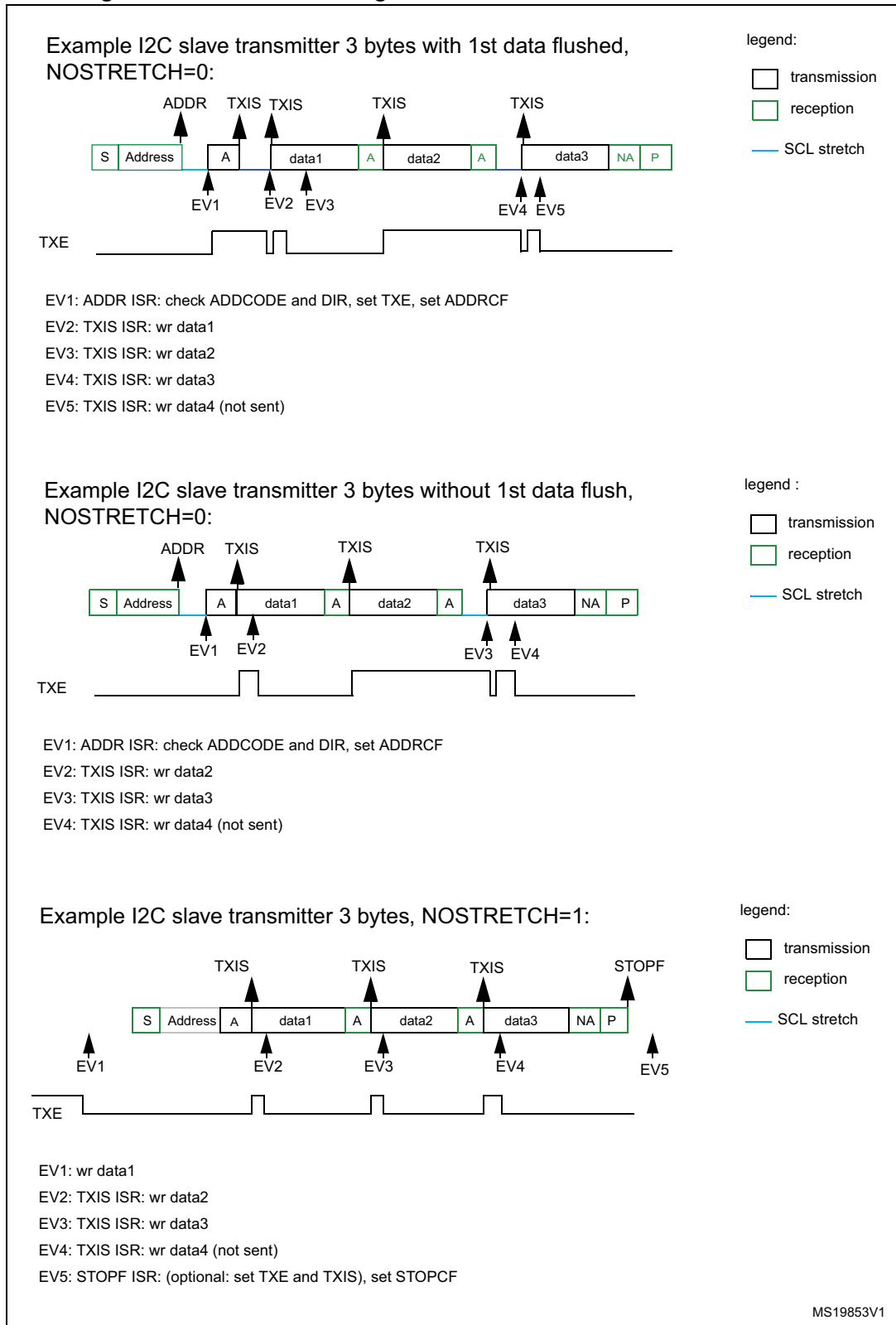


Figure 430. Transfer bus diagrams for I2C slave transmitter



**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 431. Transfer sequence flowchart for slave receiver with NOSTRETCH=0**

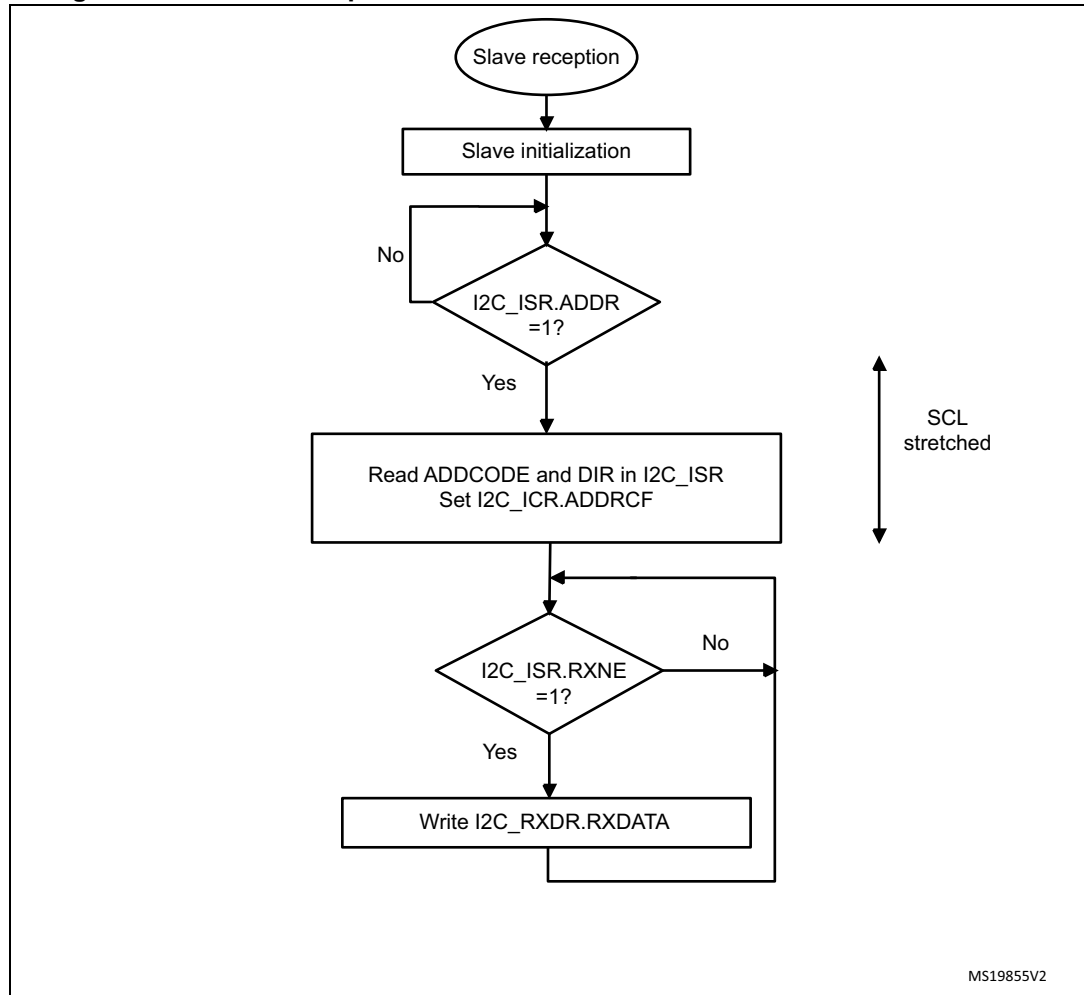


Figure 432. Transfer sequence flowchart for slave receiver with NOSTRETCH=1

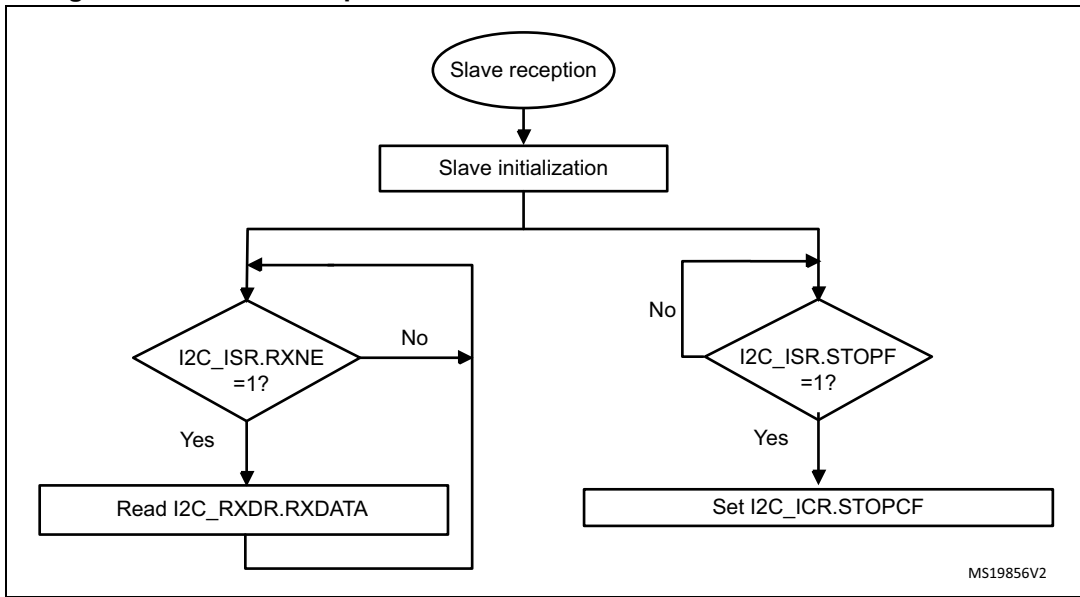
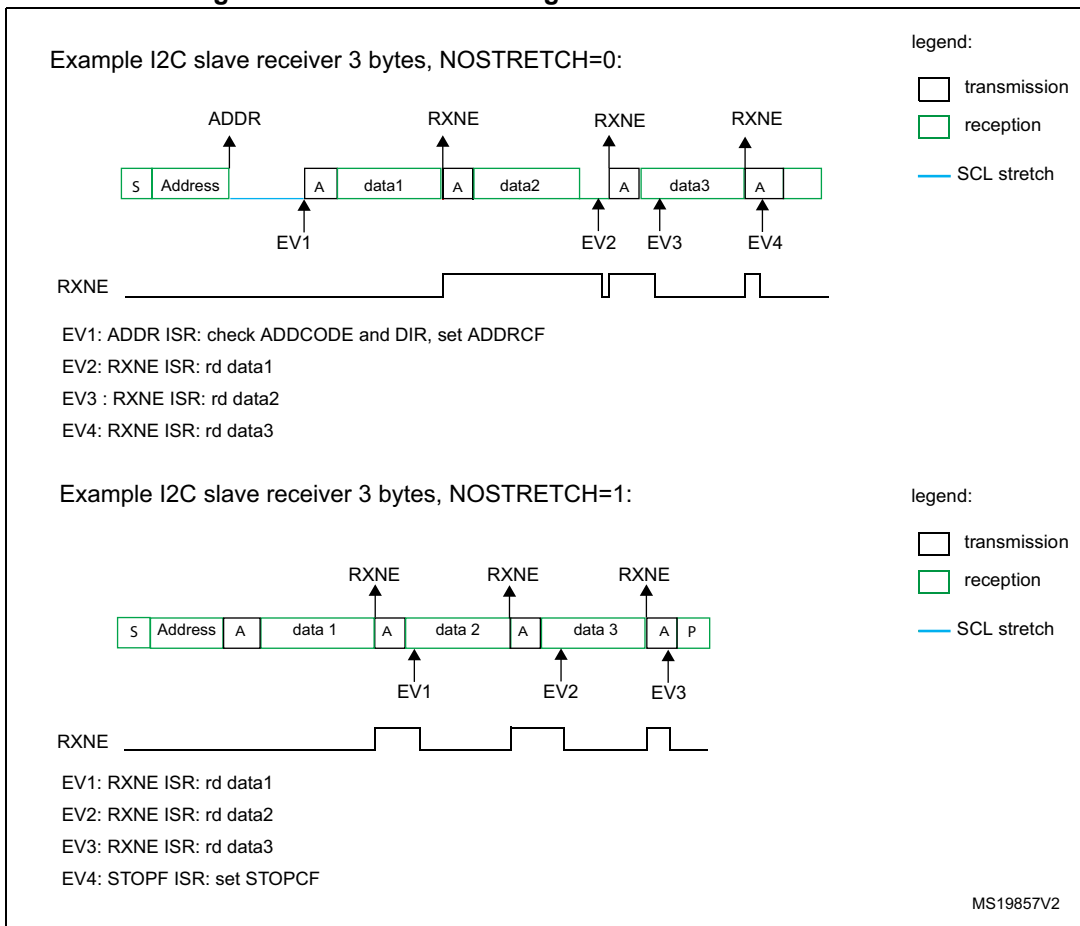


Figure 433. Transfer bus diagrams for I2C slave receiver





### 43.4.8 I2C master mode

#### I2C master initialization

Before enabling the peripheral, the I2C master clock must be configured by setting the SCLH and SCLL bits in the I2C\_TIMINGR register.

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  $t_{\text{SYNC1}}$  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  $t_{\text{SYNC2}}$  delay depending on the SCL rising edge, SCL input noise filters (analog + digital) and SCL synchronization to I2CxCLK clock. The I2C ties SCL to low level once the SCLH counter is reached reaches the value programmed in the SCLH[7:0] bits in the I2C\_TIMINGR register.

Consequently the master clock period is:

$$t_{\text{SCL}} = t_{\text{SYNC1}} + t_{\text{SYNC2}} + \{[(\text{SCLH}+1) + (\text{SCLL}+1)] \times (\text{PRESC}+1) \times t_{\text{I2CCLK}}\}$$

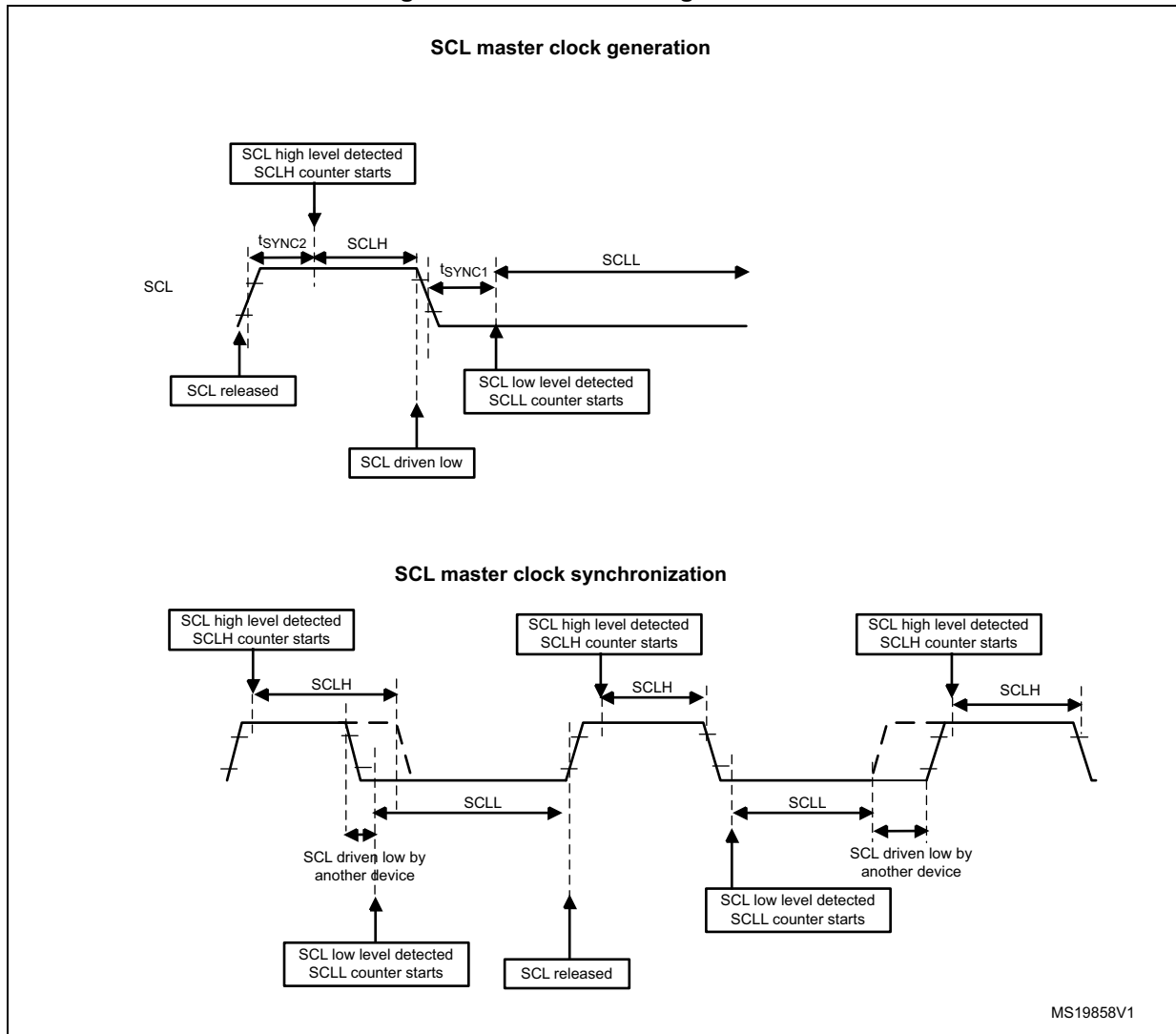
The duration of  $t_{\text{SYNC1}}$  depends on these parameters:

- SCL falling slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter:  $\text{DNF} \times t_{\text{I2CCLK}}$
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

The duration of  $t_{\text{SYNC2}}$  depends on these parameters:

- SCL rising slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter:  $\text{DNF} \times t_{\text{I2CCLK}}$
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

Figure 434. Master clock generation



**Caution:** In order to be I<sup>2</sup>C or SMBus compliant, the master clock must respect the timings given below:

Table 252. I<sup>2</sup>C-SMBUS specification clock timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBUS		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f <sub>SCL</sub>	SCL clock frequency	-	100	-	400	-	1000	-	100	kHz
t <sub>HD:STA</sub>	Hold time (repeated) START condition	4.0	-	0.6	-	0.26	-	4.0	-	μs
t <sub>SU:STA</sub>	Set-up time for a repeated START condition	4.7	-	0.6	-	0.26	-	4.7	-	μs
t <sub>SU:STO</sub>	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	μs
t <sub>BUF</sub>	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	μs
t <sub>LOW</sub>	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	μs
t <sub>HIGH</sub>	Period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	μs
t <sub>r</sub>	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t <sub>f</sub>	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	ns

Note: SCLL is also used to generate the t<sub>BUF</sub> and t<sub>SU:STA</sub> timings.

SCLH is also used to generate the t<sub>HD:STA</sub> and t<sub>SU:STO</sub> timings.

Refer to [Section 43.4.9: I2C\\_TIMINGR register configuration examples](#) for examples of I2C\_TIMINGR settings vs. I2CCLK frequency.

### Master communication initialization (address phase)

In order to initiate the communication, the user must program the following parameters for the addressed slave in the I2C\_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Slave address to be sent: SADD[9:0]
- Transfer direction: RD\_WRN
- In case of 10-bit address read: HEAD10R bit. HEAD10R must be configure to indicate if the complete address sequence must be sent, or only the header in case of a direction change.
- The number of bytes to be transferred: NBYTES[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.

The user must then set the START bit in I2C\_CR2 register. Changing all the above bits is not allowed when START bit is set.

Then the master automatically sends the START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after a delay of t<sub>BUF</sub>.

In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

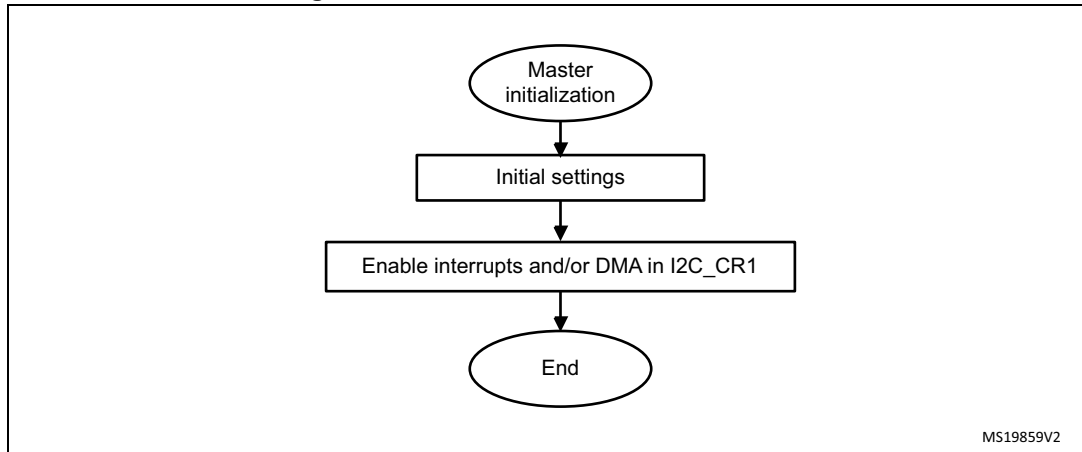
Note: The START bit is reset by hardware when the slave address has been sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware if an arbitration loss occurs.  
In 10-bit addressing mode, when the Slave Address first 7 bits is NACKed by the slave, the

master will re-launch 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 435. Master initialization flowchart

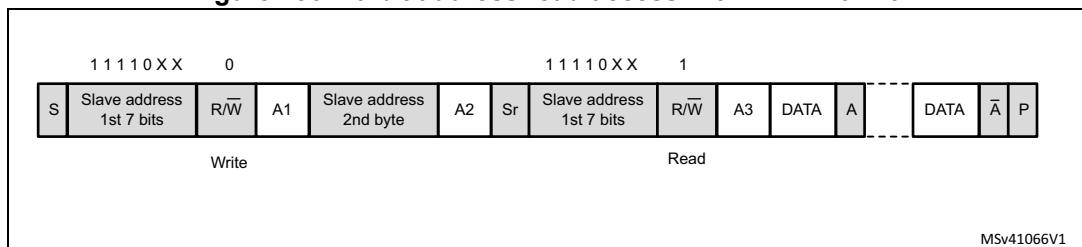


MS19859V2

**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 2nd byte + REStart + Slave address 10-bit header Read

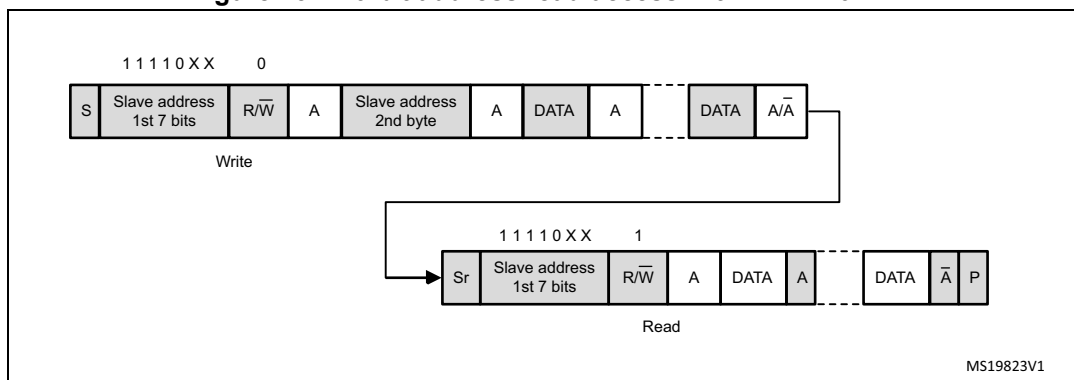
Figure 436. 10-bit address read access with HEAD10R=0



MSv41066V1

- 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 437. 10-bit address read access with HEAD10R=1**



**Master transmitter**

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the 9th SCL pulse when an ACK is received.

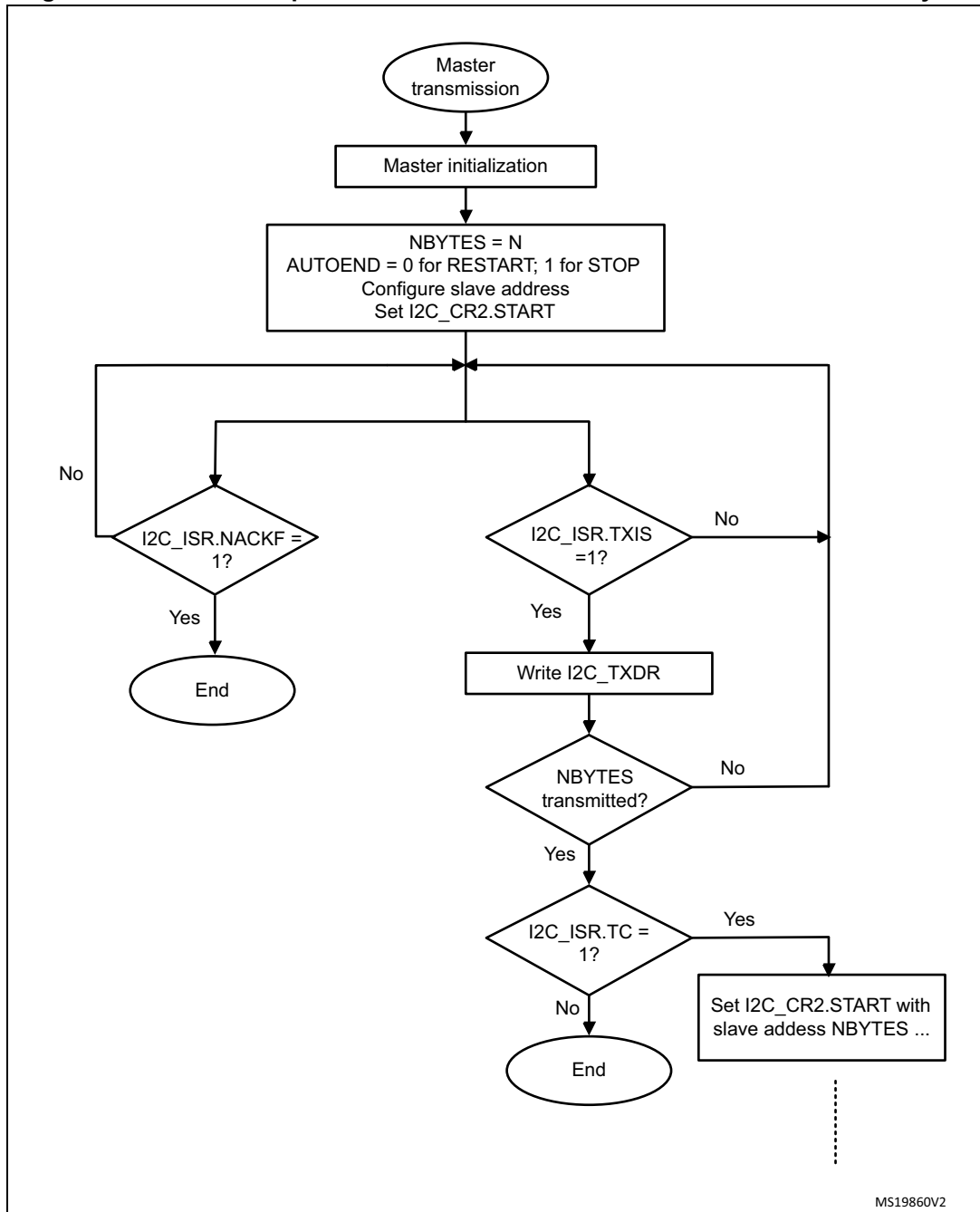
A TXIS event generates an interrupt if the TXIE bit is set in the I2C\_CR1 register. The flag is cleared when the I2C\_TXDR register is written with the next data byte to be transmitted.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to be sent is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C\_CR2 register. In this case, when NBYTES data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

The TXIS flag is not set when a NACK is received.

- When RELOAD=0 and NBYTES data have been transferred:
  - In automatic end mode (AUTOEND=1), a STOP is automatically sent.
  - In software end mode (AUTOEND=0), the TC flag is set and the SCL line is stretched low in order to perform software actions:
    - A RESTART condition can be requested by setting the START bit in the I2C\_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition is sent on the bus.
    - A STOP condition can be requested by setting the STOP bit in the I2C\_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.
- If a NACK is received: the TXIS flag is not set, and a STOP condition is automatically sent after the NACK reception. the NACKF flag is set in the I2C\_ISR register, and an interrupt is generated if the NACKIE bit is set.

Figure 438. Transfer sequence flowchart for I2C master transmitter for N≤255 bytes



MS19860V2

Figure 439. Transfer sequence flowchart for I2C master transmitter for N>255 bytes

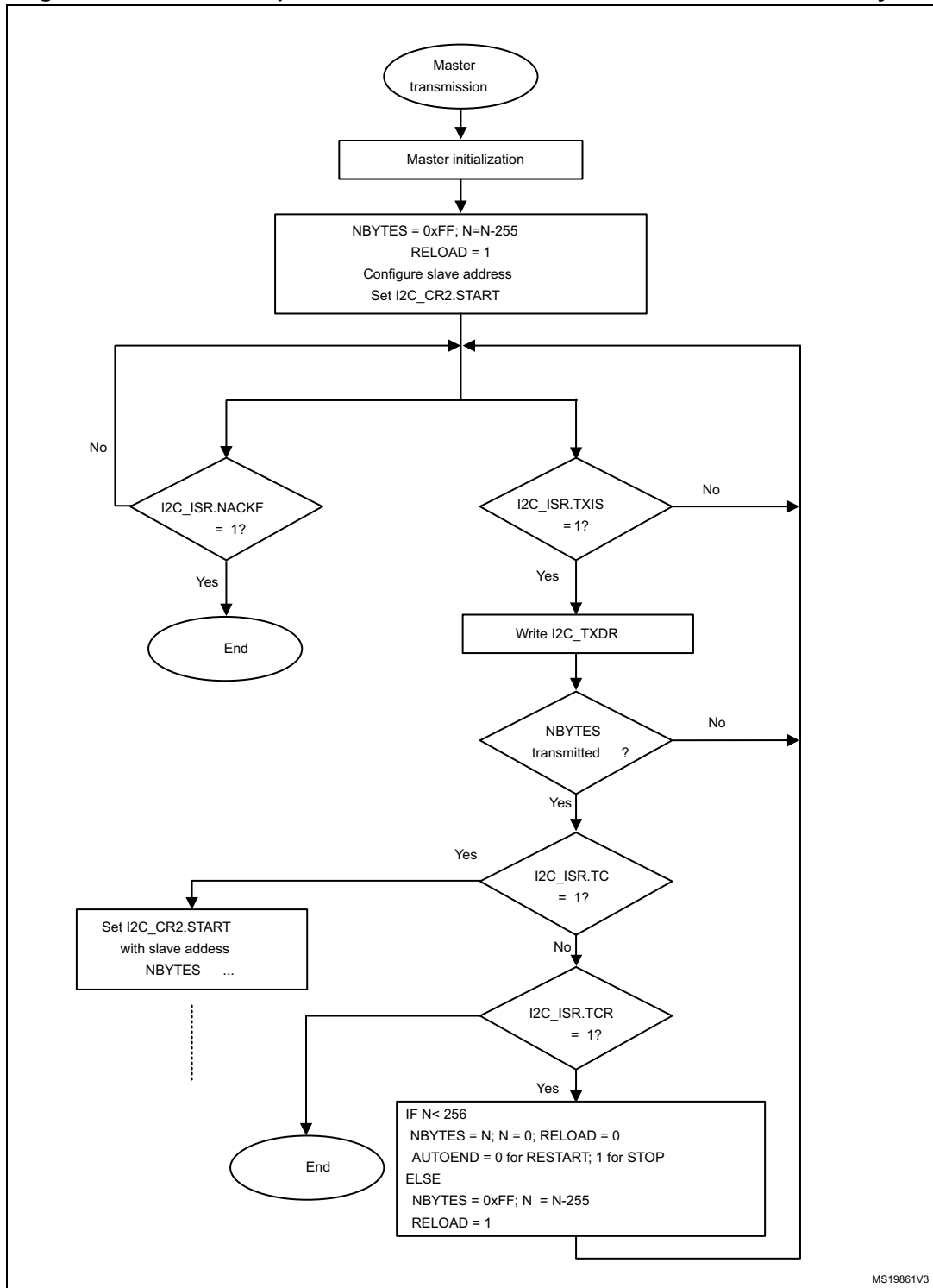
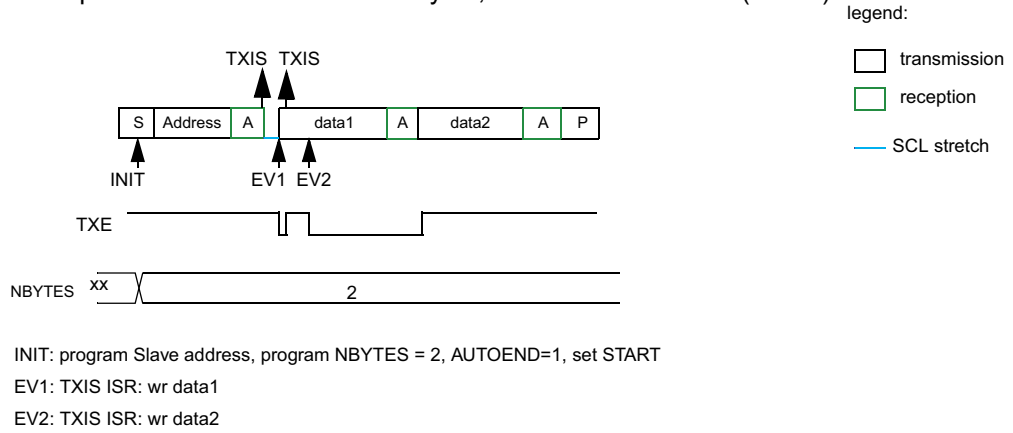
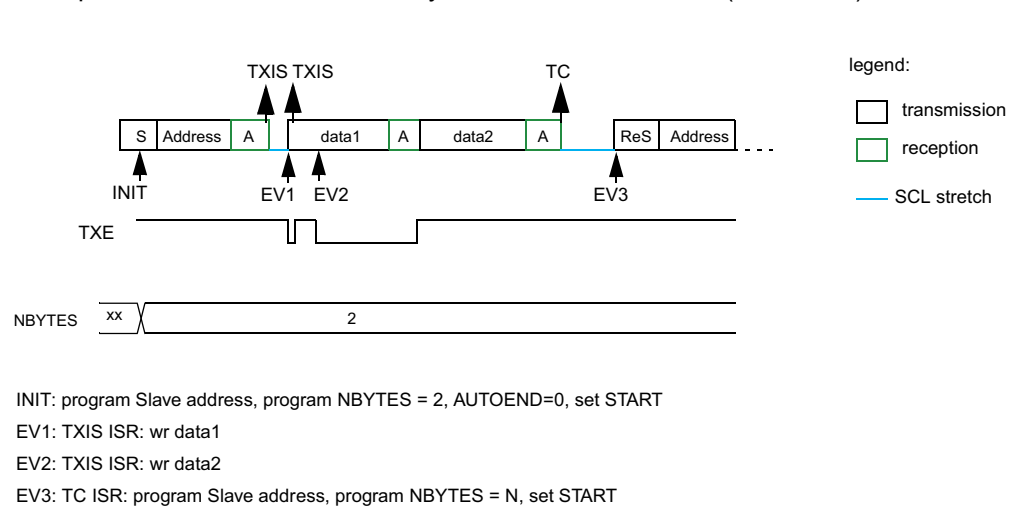


Figure 440. Transfer bus diagrams for I2C master transmitter

Example I2C master transmitter 2 bytes, automatic end mode (STOP)



Example I2C master transmitter 2 bytes, software end mode (RESTART)



MS19862V1



### Master receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the 8th SCL pulse. An RXNE event generates an interrupt if the RXIE bit is set in the I2C\_CR1 register. The flag is cleared when I2C\_RXDR is read.

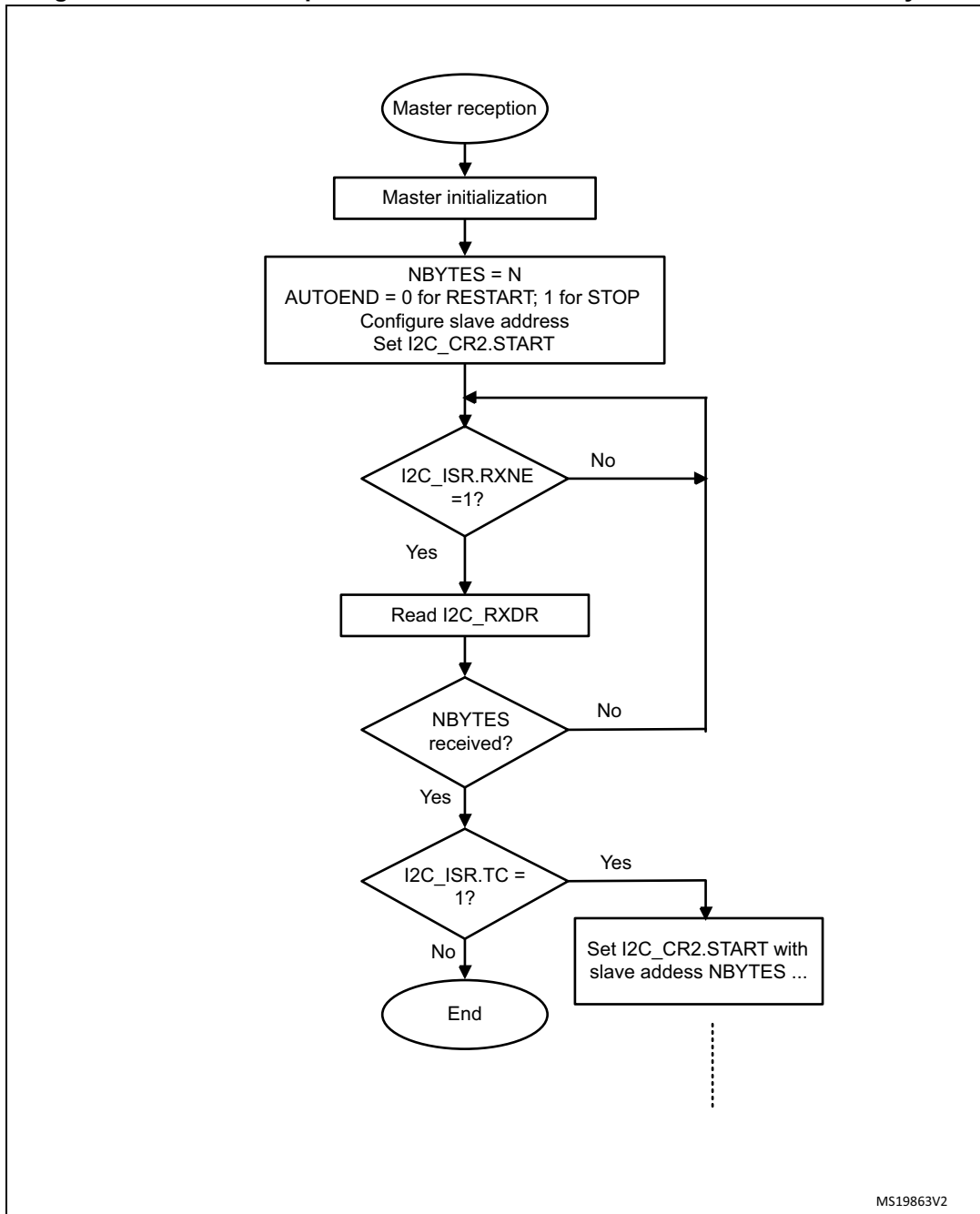
If the total number of data bytes to be received is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C\_CR2 register. In this case, when NBYTES[7:0] data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

- When RELOAD=0 and NBYTES[7:0] data have been transferred:
  - In automatic end mode (AUTOEND=1), a NACK and a STOP are automatically sent after the last received byte.
  - In software end mode (AUTOEND=0), a NACK is automatically sent after the last received byte, the TC flag is set and the SCL line is stretched low in order to allow software actions:

A RESTART condition can be requested by setting the START bit in the I2C\_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition, followed by slave address, are sent on the bus.

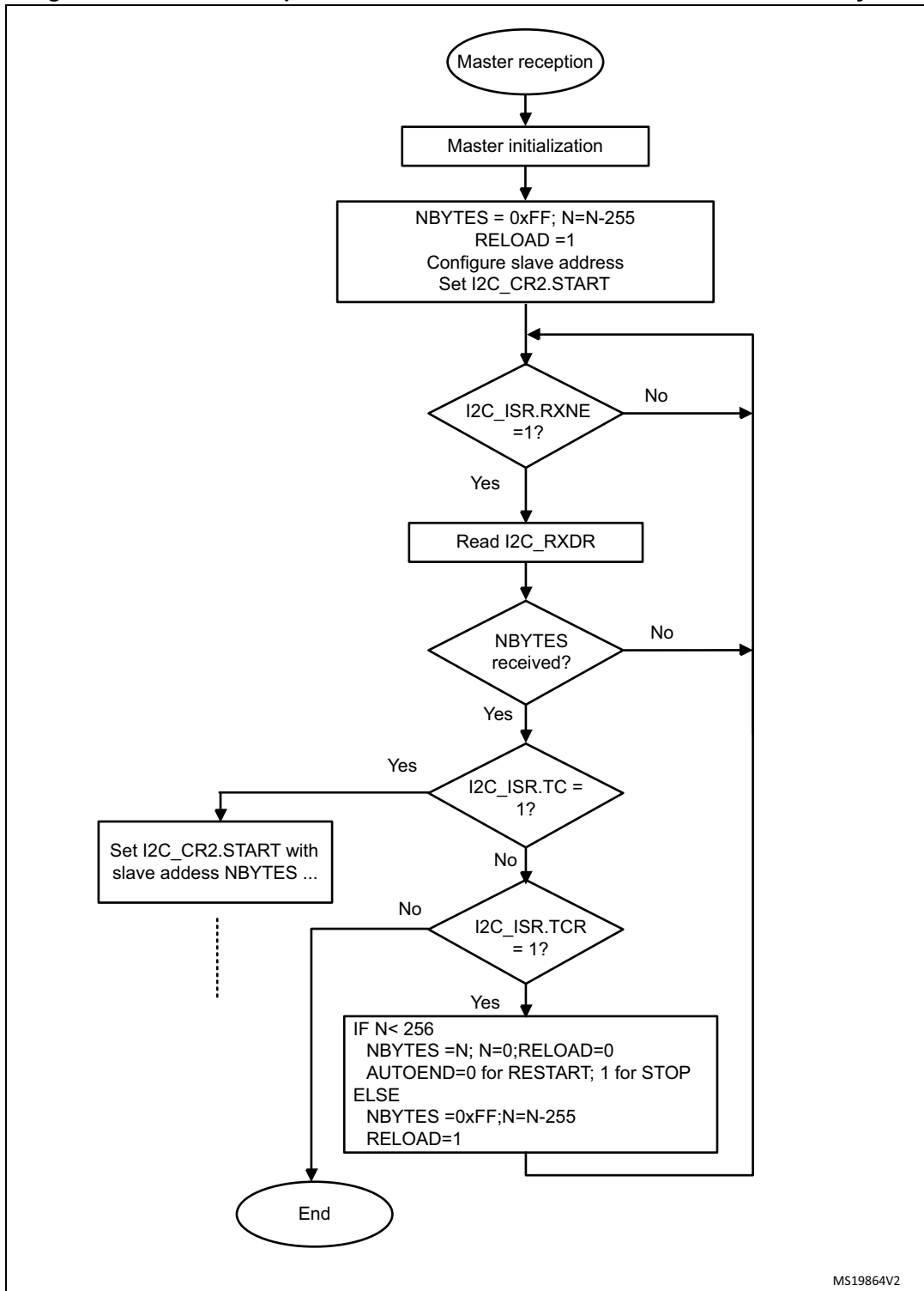
A STOP condition can be requested by setting the STOP bit in the I2C\_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.

Figure 441. Transfer sequence flowchart for I2C master receiver for N≤255 bytes



MS19863V2

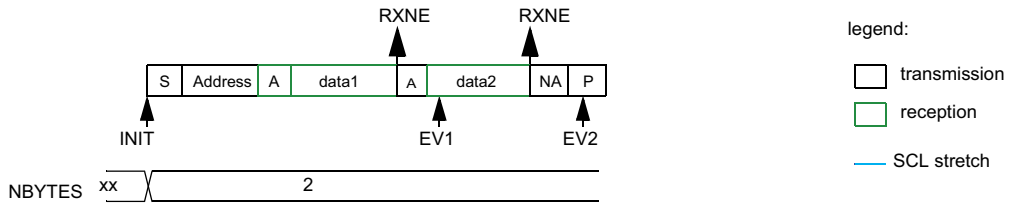
Figure 442. Transfer sequence flowchart for I2C master receiver for N >255 bytes



MS19864V2

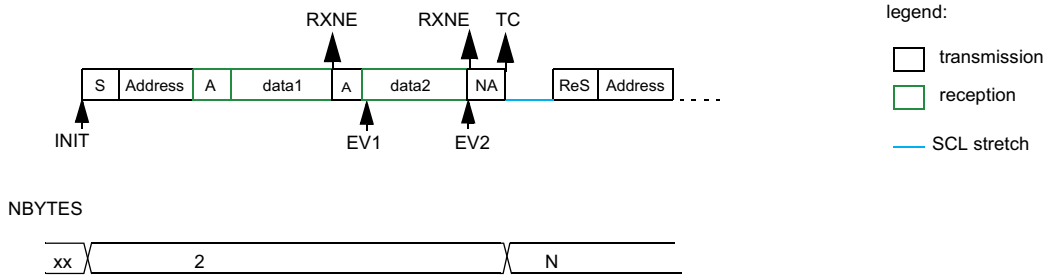
Figure 443. Transfer bus diagrams for I2C master receiver

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

### 43.4.9 I2C\_TIMINGR register configuration examples

The tables below provide examples of how to program the I2C\_TIMINGR to obtain timings compliant with the I<sup>2</sup>C specification. In order to get more accurate configuration values, the STM32CubeMX tool (I2C Configuration window) should be used.

**Table 253. 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
$t_{SCLL}$	200x250 ns = 50 $\mu$ s	20x250 ns = 5.0 $\mu$ s	10x125 ns = 1250 ns	7x125 ns = 875 ns
SCLH	0xC3	0xF	0x3	0x3
$t_{SCLH}$	196x250 ns = 49 $\mu$ s	16x250 ns = 4.0 $\mu$ s	4x125ns = 500ns	4x125 ns = 500 ns
$t_{SCL}^{(1)}$	~100 $\mu$ s <sup>(2)</sup>	~10 $\mu$ s <sup>(2)</sup>	~2500 ns <sup>(3)</sup>	~2000 ns <sup>(4)</sup>
SDADEL	0x2	0x2	0x1	0x0
$t_{SDADEL}$	2x250 ns = 500 ns	2x250 ns = 500 ns	1x125 ns = 125 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
$t_{SCLDEL}$	5x250 ns = 1250 ns	5x250 ns = 1250 ns	4x125 ns = 500 ns	2x125 ns = 250 ns

1. SCL period  $t_{SCL}$  is greater than  $t_{SCLL} + t_{SCLH}$  due to SCL internal detection delay. Values provided for  $t_{SCL}$  are examples only.
2.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 500$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 1000$  ns
3.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 500$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 750$  ns
4.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 500$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 655$  ns

**Table 254. 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 x 250 ns = 50 $\mu$ s	20 x 250 ns = 5.0 $\mu$ s	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH	0xC3	0xF	0x3	0x2
$t_{SCLH}$	196 x 250 ns = 49 $\mu$ s	16 x 250 ns = 4.0 $\mu$ s	4 x 125ns = 500 ns	3 x 62.5 ns = 187.5 ns
$t_{SCL}^{(1)}$	~100 $\mu$ s <sup>(2)</sup>	~10 $\mu$ s <sup>(2)</sup>	~2500 ns <sup>(3)</sup>	~1000 ns <sup>(4)</sup>
SDADEL	0x2	0x2	0x2	0x0
$t_{SDADEL}$	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x2
$t_{SCLDEL}$	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. SCL period  $t_{SCL}$  is greater than  $t_{SCLL} + t_{SCLH}$  due to SCL internal detection delay. Values provided for  $t_{SCL}$  are examples only.

2.  $t_{\text{SYNC1}} + t_{\text{SYNC2}}$  minimum value is  $4 \times t_{\text{I2CCLK}} = 250$  ns. Example with  $t_{\text{SYNC1}} + t_{\text{SYNC2}} = 1000$  ns
3.  $t_{\text{SYNC1}} + t_{\text{SYNC2}}$  minimum value is  $4 \times t_{\text{I2CCLK}} = 250$  ns. Example with  $t_{\text{SYNC1}} + t_{\text{SYNC2}} = 750$  ns
4.  $t_{\text{SYNC1}} + t_{\text{SYNC2}}$  minimum value is  $4 \times t_{\text{I2CCLK}} = 250$  ns. Example with  $t_{\text{SYNC1}} + t_{\text{SYNC2}} = 500$  ns

Table 255. Examples of timings settings for  $f_{\text{I2CCLK}} = 48$  MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	0xB	0xB	5	5
SCLL	0xC7	0x13	0x9	0x3
$t_{\text{SCLL}}$	$200 \times 250$ ns = 50 $\mu$ s	$20 \times 250$ ns = 5.0 $\mu$ s	$10 \times 125$ ns = 1250 ns	$4 \times 125$ ns = 500 ns
SCLH	0xC3	0xF	0x3	0x1
$t_{\text{SCLH}}$	$196 \times 250$ ns = 49 $\mu$ s	$16 \times 250$ ns = 4.0 $\mu$ s	$4 \times 125$ ns = 500 ns	$2 \times 125$ ns = 250 ns
$t_{\text{SCL}}^{(1)}$	$\sim 100$ $\mu$ s <sup>(2)</sup>	$\sim 10$ $\mu$ s <sup>(2)</sup>	$\sim 2500$ ns <sup>(3)</sup>	$\sim 875$ ns <sup>(4)</sup>
SDADEL	0x2	0x2	0x3	0x0
$t_{\text{SDADEL}}$	$2 \times 250$ ns = 500 ns	$2 \times 250$ ns = 500 ns	$3 \times 125$ ns = 375 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
$t_{\text{SCLDEL}}$	$5 \times 250$ ns = 1250 ns	$5 \times 250$ ns = 1250 ns	$4 \times 125$ ns = 500 ns	$2 \times 125$ ns = 250 ns

1. The SCL period  $t_{\text{SCL}}$  is greater than  $t_{\text{SCLL}} + t_{\text{SCLH}}$  due to the SCL internal detection delay. Values provided for  $t_{\text{SCL}}$  are only examples.
2.  $t_{\text{SYNC1}} + t_{\text{SYNC2}}$  minimum value is  $4 \times t_{\text{I2CCLK}} = 83.3$  ns. Example with  $t_{\text{SYNC1}} + t_{\text{SYNC2}} = 1000$  ns
3.  $t_{\text{SYNC1}} + t_{\text{SYNC2}}$  minimum value is  $4 \times t_{\text{I2CCLK}} = 83.3$  ns. Example with  $t_{\text{SYNC1}} + t_{\text{SYNC2}} = 750$  ns
4.  $t_{\text{SYNC1}} + t_{\text{SYNC2}}$  minimum value is  $4 \times t_{\text{I2CCLK}} = 83.3$  ns. Example with  $t_{\text{SYNC1}} + t_{\text{SYNC2}} = 250$  ns

### 43.4.10 SMBus specific features

This section is relevant only when SMBus feature is supported. Refer to [Section 43.3: 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. 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 should be implemented by the user software.

For more details of 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 should be implemented by the user software.

Arbitration is also performed in slave mode for ARP support.

For more details of the SMBus Address Resolution Protocol, refer to SMBus specification (<http://smbus.org>).

## Received Command and Data acknowledge control

A SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in slave mode, the Slave Byte Control mode must be enabled by setting SBC bit in I2C\_CR1 register. Refer to [Slave Byte Control mode on page 1460](#) for more details.

## Host Notify protocol

This peripheral supports the Host Notify protocol by setting the SMBHEN bit in the I2C\_CR1 register. In this case the host will acknowledge the SMBus Host address (0b0001 000).

When this protocol is used, the device acts as a master and the host as a slave.

## SMBus alert

The SMBus ALERT optional signal is supported. A slave-only device can signal the host through the SMBALERT# pin that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the Alert Response Address (0b0001 100). Only the device(s) which pulled SMBALERT# low will acknowledge the Alert Response Address.

When configured as a slave device (SMBHEN=0), the SMBA pin is pulled low by setting the ALERTEN bit in the I2C\_CR1 register. The Alert Response Address is enabled at the same time.

When configured as a host (SMBHEN=1), the ALERT flag is set in the I2C\_ISR register when a falling edge is detected on the SMBA pin and ALERTEN=1. An interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register. When ALERTEN=0, the ALERT line is considered high even if the external SMBA pin is low.

*If the SMBus ALERT pin is not needed, the SMBA pin can be used as a standard GPIO if ALERTEN=0.*

### Packet error checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. Packet Error Checking is implemented by appending a Packet Error Code (PEC) at the end of each message transfer. The PEC is calculated by using the  $C(x) = x^8 + x^2 + x + 1$  CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator and allows to send a Not Acknowledge automatically when the received byte does not match with the hardware calculated PEC.

### Timeouts

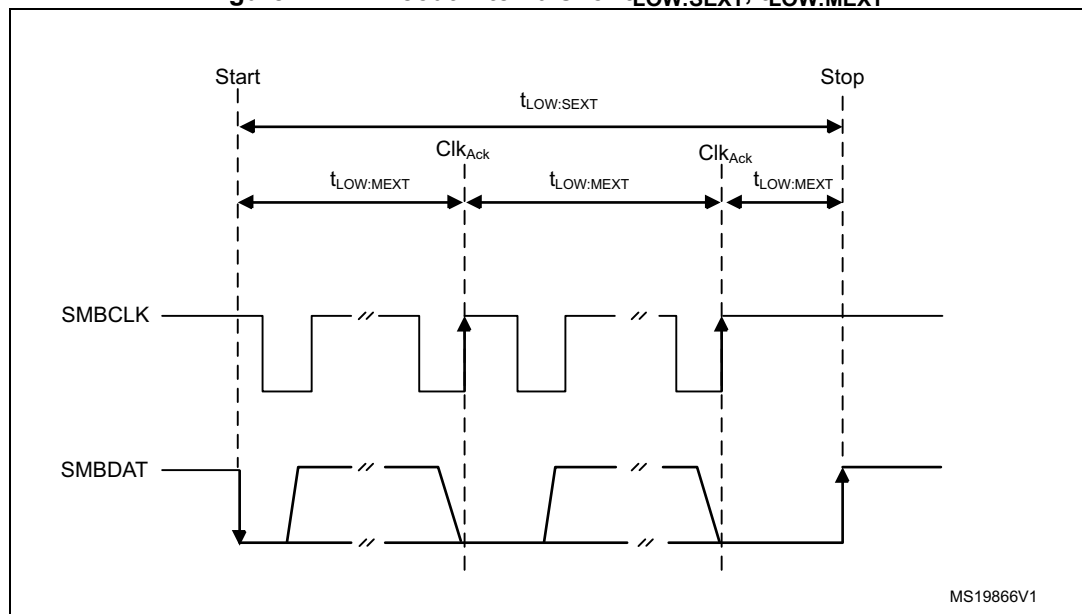
This peripheral embeds hardware timers in order to be compliant with the 3 timeouts defined in SMBus specification.

**Table 256. SMBus timeout specifications**

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{\text{TIMEOUT}}$	Detect clock low timeout	25	35	ms
$t_{\text{LOW:SEXT}}^{(1)}$	Cumulative clock low extend time (slave device)	-	25	ms
$t_{\text{LOW:MEXT}}^{(2)}$	Cumulative clock low extend time (master device)	-	10	ms

- $t_{\text{LOW:SEXT}}$  is the cumulative time a given slave device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that, another slave device or the master will also extend the clock causing the combined clock low extend time to be greater than  $t_{\text{LOW:SEXT}}$ . Therefore, this parameter is measured with the slave device as the sole target of a full-speed master.
- $t_{\text{LOW:MEXT}}$  is the cumulative time a master device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a slave device or another master will also extend the clock causing the combined clock low time to be greater than  $t_{\text{LOW:MEXT}}$  on a given byte. Therefore, this parameter is measured with a full speed slave device as the sole target of the master.

**Figure 444. Timeout intervals for  $t_{\text{LOW:SEXT}}$ ,  $t_{\text{LOW:MEXT}}$**



MS19866V1



### 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 250: I2C-SMBUS specification data setup and hold times](#))

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.

## 43.4.11 SMBus initialization

This section is relevant only when SMBus feature is supported. Refer to [Section 43.3: I2C implementation](#).

In addition to I2C initialization, some other specific initialization must be done in order to perform SMBus communication:

### Received Command and Data Acknowledge control (Slave mode)

A SMBus receiver must be able to NACK each received command or data. In order to allow ACK control in slave mode, the Slave Byte Control mode must be enabled by setting the SBC bit in the I2C\_CR1 register. Refer to [Slave Byte Control mode on page 1460](#) for more details.

### Specific address (Slave mode)

The specific SMBus addresses should be enabled if needed. Refer to [Bus idle detection on page 1483](#) 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 257. 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\_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

- $t_{\text{TIMEOUT}}$  check  
 In order to enable the  $t_{\text{TIMEOUT}}$  check, the 12-bit TIMEOUTA[11:0] bits must be programmed with the timer reload value in order to check the  $t_{\text{TIMEOUT}}$  parameter. The TIDLE bit must be configured to '0' in order to detect the SCL low level timeout.  
 Then the timer is enabled by setting the TIMOUTEN in the I2C\_TIMEOUTR register.  
 If SCL is tied low for a time greater than  $(\text{TIMEOUTA}+1) \times 2048 \times t_{\text{I2CCLK}}$ , the TIMEOUT flag is set in the I2C\_ISR register.  
 Refer to [Table 258: Examples of TIMEOUTA settings for various I2CCLK frequencies \(max  \$t\_{\text{TIMEOUT}} = 25 \text{ ms}\$ \)](#).

**Caution:** Changing the TIMEOUTA[11:0] bits and TIDLE bit configuration is not allowed when the TIMOUTEN bit is set.

- $t_{\text{LOW:SEXT}}$  and  $t_{\text{LOW:MEXT}}$  check  
 Depending on if the peripheral is configured as a master or as a slave, The 12-bit TIMEOUTB timer must be configured in order to check  $t_{\text{LOW:SEXT}}$  for a slave and  $t_{\text{LOW:MEXT}}$  for a master. As the standard specifies only a maximum, the user can choose the same value for the both.  
 Then the timer is enabled by setting the TEXTEN bit in the I2C\_TIMEOUTR register.  
 If the SMBus peripheral performs a cumulative SCL stretch for a time greater than  $(\text{TIMEOUTB}+1) \times 2048 \times t_{\text{I2CCLK}}$ , and in the timeout interval described in [Bus idle detection on page 1483](#) section, the TIMEOUT flag is set in the I2C\_ISR register.  
 Refer to [Table 259: Examples of TIMEOUTB settings for various I2CCLK frequencies](#)

**Caution:** Changing the TIMEOUTB configuration is not allowed when the TEXTEN bit is set.

### Bus Idle detection

In order to enable the  $t_{\text{IDLE}}$  check, the 12-bit TIMEOUTA[11:0] field must be programmed with the timer reload value in order to obtain the  $t_{\text{IDLE}}$  parameter. The TIDLE bit must be configured to '1' in order to detect both SCL and SDA high level timeout.

Then the timer is enabled by setting the TIMOUTEN bit in the I2C\_TIMEOUTR register.

If both the SCL and SDA lines remain high for a time greater than  $(\text{TIMEOUTA}+1) \times 4 \times t_{\text{I2CCLK}}$ , the TIMEOUT flag is set in the I2C\_ISR register.

Refer to [Table 260: Examples of TIMEOUTA settings for various I2CCLK frequencies \(max  \$t\_{\text{IDLE}} = 50 \mu\text{s}\$ \)](#)

**Caution:** Changing the TIMEOUTA and TIDLE configuration is not allowed when the TIMEOUTEN is set.

#### 43.4.12 SMBus: I2C\_TIMEOUTR register configuration examples

This section is relevant only when SMBus feature is supported. Refer to [Section 43.3: I2C implementation](#).

- Configuring the maximum duration of  $t_{\text{TIMEOUT}}$  to 25 ms:

**Table 258. Examples of TIMEOUTA settings for various I2CCLK frequencies (max  $t_{\text{TIMEOUT}} = 25$  ms)**

$f_{\text{I2CCLK}}$	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	$t_{\text{TIMEOUT}}$
8 MHz	0x61	0	1	$98 \times 2048 \times 125 \text{ ns} = 25 \text{ ms}$
16 MHz	0xC3	0	1	$196 \times 2048 \times 62.5 \text{ ns} = 25 \text{ ms}$
32 MHz	0x186	0	1	$391 \times 2048 \times 31.25 \text{ ns} = 25 \text{ ms}$
48 MHz	0x249	0	1	$586 \times 2048 \times 20.08 \text{ ns} = 25 \text{ ms}$

- Configuring the maximum duration of  $t_{\text{LOW:SEXT}}$  and  $t_{\text{LOW:MEXT}}$  to 8 ms:

**Table 259. Examples of TIMEOUTB settings for various I2CCLK frequencies**

$f_{\text{I2CCLK}}$	TIMEOUTB[11:0] bits	TEXTEN bit	$t_{\text{LOW:EXT}}$
8 MHz	0x1F	1	$32 \times 2048 \times 125 \text{ ns} = 8 \text{ ms}$
16 MHz	0x3F	1	$64 \times 2048 \times 62.5 \text{ ns} = 8 \text{ ms}$
48 MHz	0xBB	1	$188 \times 2048 \times 20.08 \text{ ns} = 8 \text{ ms}$

- Configuring the maximum duration of  $t_{\text{IDLE}}$  to 50  $\mu\text{s}$

**Table 260. Examples of TIMEOUTA settings for various I2CCLK frequencies (max  $t_{\text{IDLE}} = 50$   $\mu\text{s}$ )**

$f_{\text{I2CCLK}}$	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	$t_{\text{TIDLE}}$
8 MHz	0x63	1	1	$100 \times 4 \times 125 \text{ ns} = 50 \mu\text{s}$
16 MHz	0xC7	1	1	$200 \times 4 \times 62.5 \text{ ns} = 50 \mu\text{s}$
48 MHz	0x257	1	1	$600 \times 4 \times 20.08 \text{ ns} = 50 \mu\text{s}$

### 43.4.13 SMBus slave mode

This section is relevant only when SMBus feature is supported. Refer to [Section 43.3: I2C implementation](#).

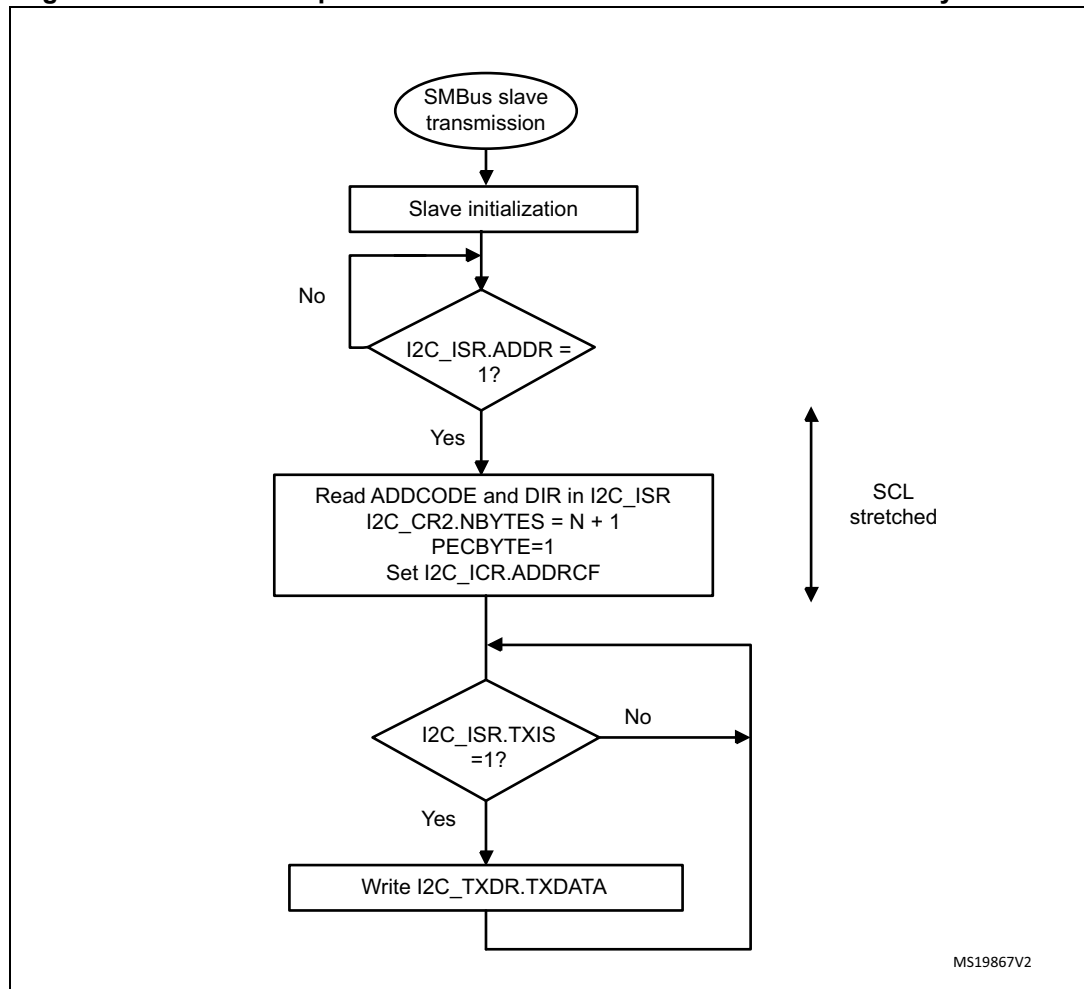
In addition to I2C slave transfer management (refer to [Section 43.4.7: I2C slave mode](#)) some additional software flowcharts are provided to support SMBus.

#### SMBus Slave transmitter

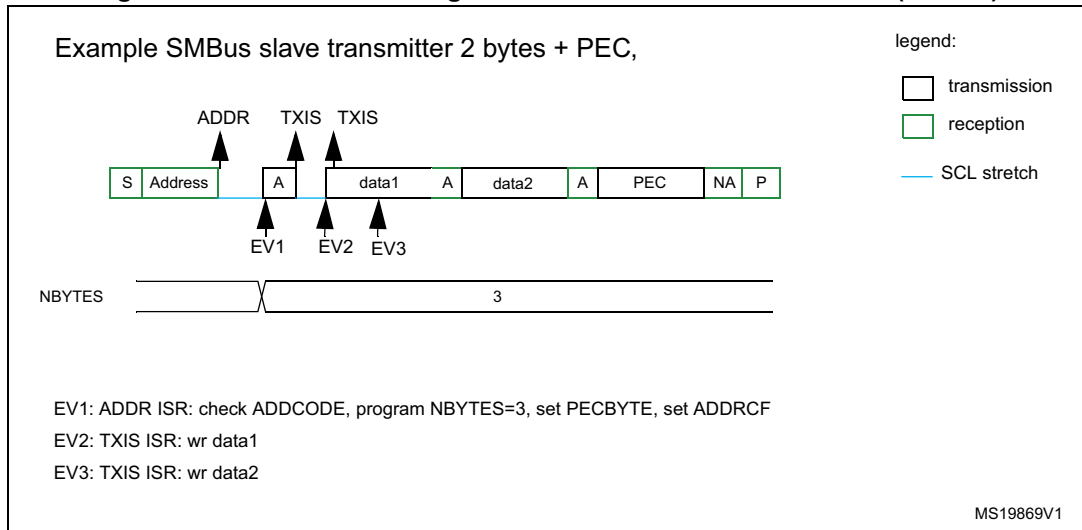
When the IP is used in SMBus, SBC must be programmed to '1' in order to allow the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case the total number of TXIS interrupts will be NBYTES-1 and the content of the I2C\_PECR register is automatically transmitted if the master requests an extra byte after the NBYTES-1 data transfer.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

**Figure 445. Transfer sequence flowchart for SMBus slave transmitter N bytes + PEC**



**Figure 446. Transfer bus diagrams for SMBus slave transmitter (SBC=1)**



**SMBus Slave receiver**

When the I2C is used in SMBus mode, SBC must be programmed to '1' in order to allow the PEC checking at the end of the programmed number of data bytes. In order to allow the ACK control of each byte, the reload mode must be selected (RELOAD=1). Refer to [Slave Byte Control mode on page 1460](#) 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 447. Transfer sequence flowchart for SMBus slave receiver N Bytes + PEC

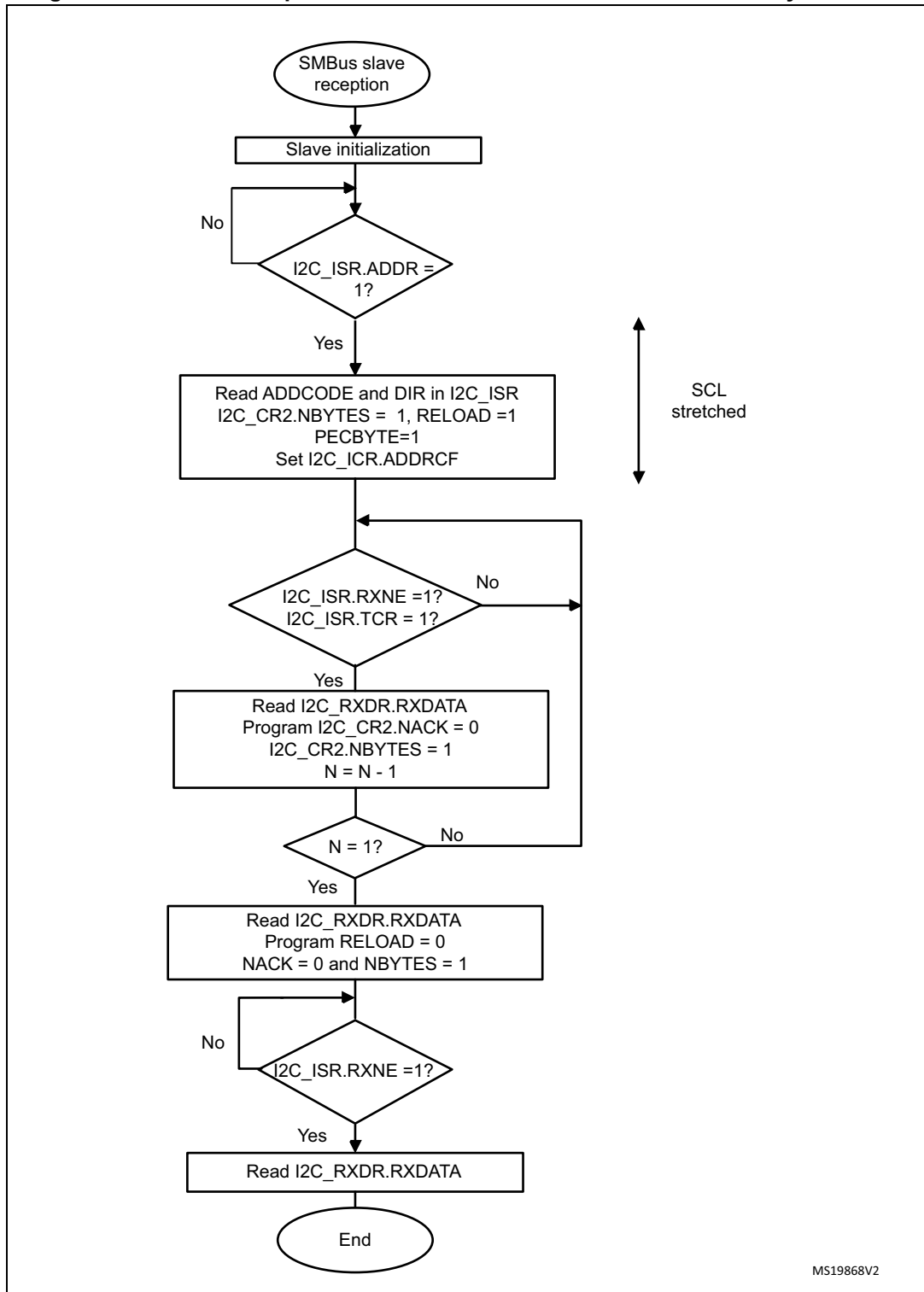
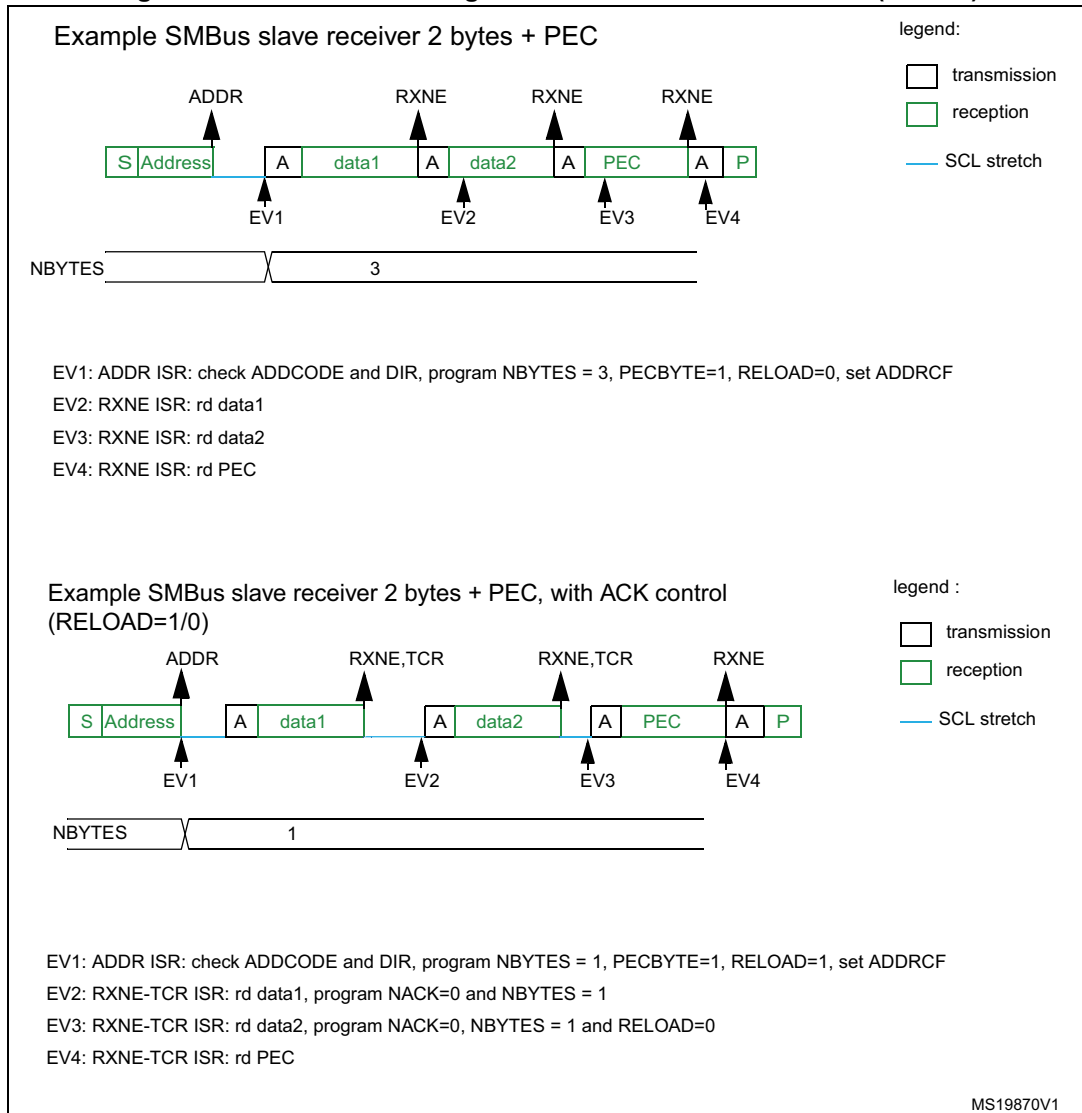


Figure 448. Bus transfer diagrams for SMBus slave receiver (SBC=1)



This section is relevant only when SMBus feature is supported. Refer to [Section 43.3: I2C implementation](#).

In addition to I2C master transfer management (refer to [Section 43.4.8: I2C master mode](#)) some additional software flowcharts are provided to support SMBus.

### SMBus Master transmitter

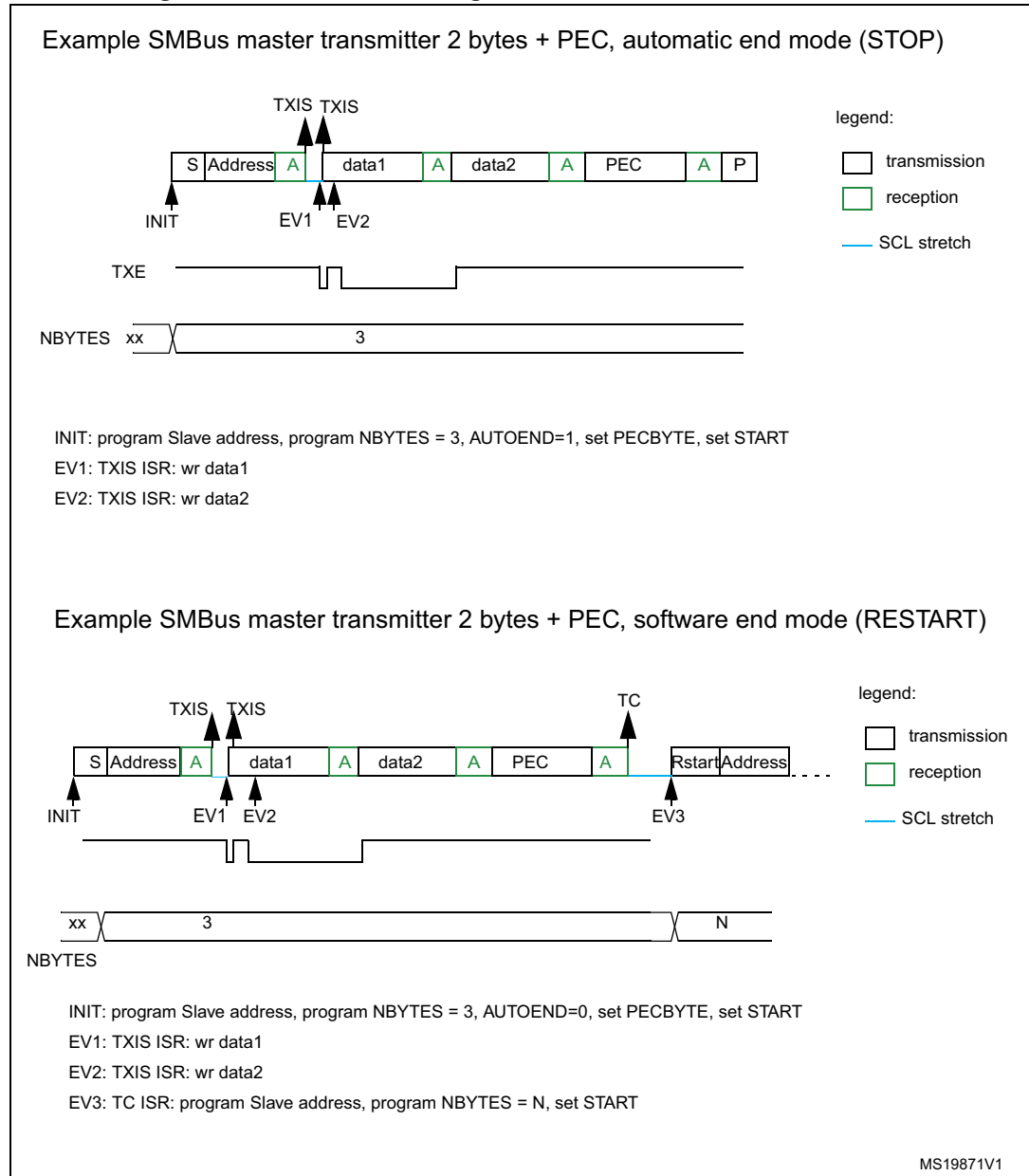
When the SMBus master wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTES[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts will be NBYTES-1. So if the PECBYTE bit is set when NBYTES=0x1, the content of the I2C\_PECR register is automatically transmitted.

If the SMBus master wants to send a STOP condition after the PEC, automatic end mode should be selected (AUTOEND=1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus master wants to send a RESTART condition after the PEC, software mode must be selected (AUTOEND=0). In this case, once NBYTES-1 have been transmitted, the I2C\_PECR register content is transmitted and the TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

**Figure 449. Bus transfer diagrams for SMBus master transmitter**





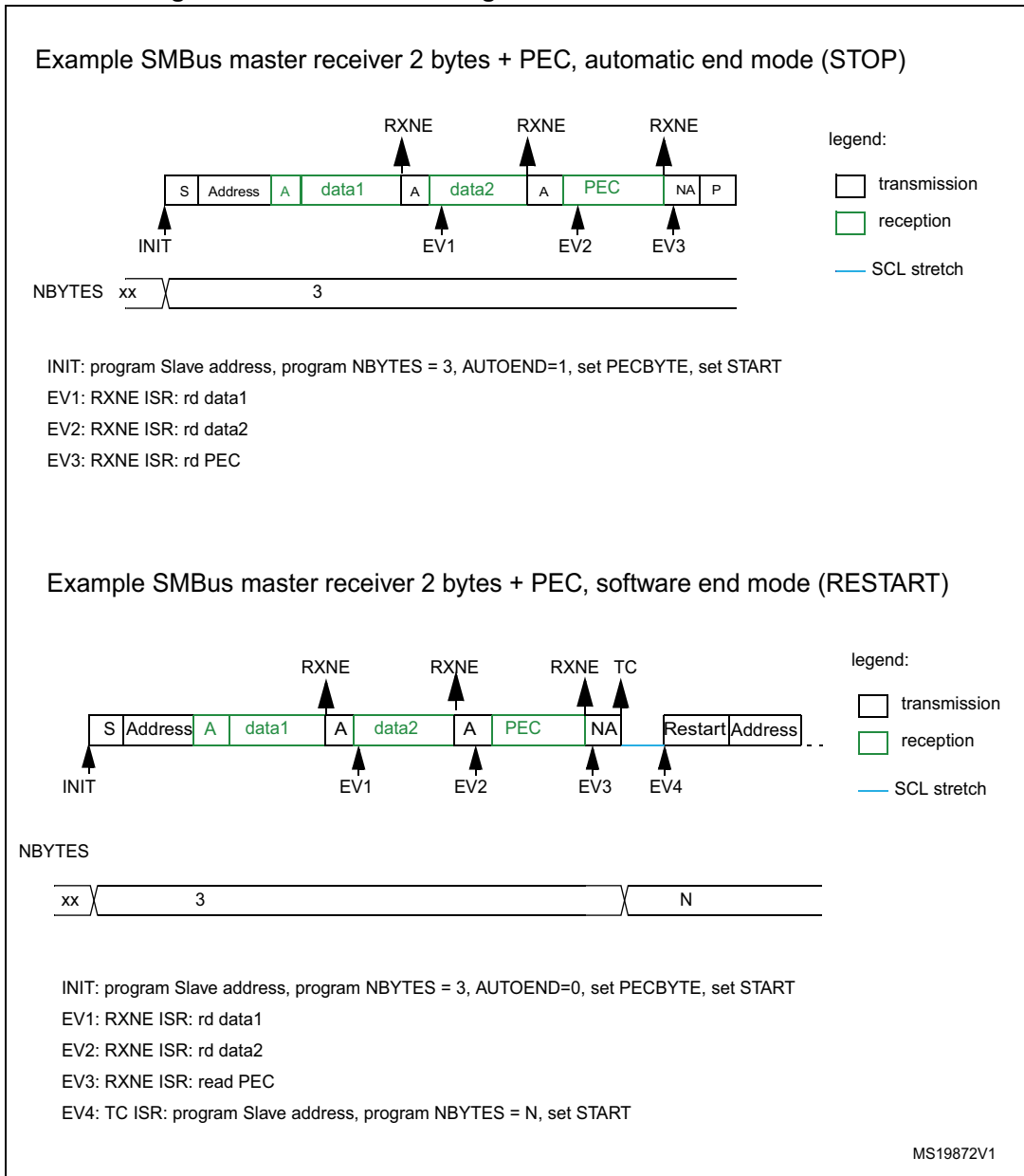
**SMBus Master receiver**

When the SMBus master wants to receive the PEC followed by a STOP at the end of the transfer, automatic end mode can be selected (AUTOEND=1). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C\_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus master receiver wants to receive the PEC byte followed by a RESTART condition at the end of the transfer, software mode must be selected (AUTOEND=0). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C\_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 450. Bus transfer diagrams for SMBus master receiver



### 43.4.14 Wakeup from Stop mode on address match

This section is relevant only when Wakeup from Stop mode feature is supported. Refer to [Section 43.3: I2C implementation](#).

The I2C is able to wakeup the MCU from Stop mode (APB clock is off), when it is addressed. All addressing modes are supported.

Wakeup from Stop mode is enabled by setting the WUPEN bit in the I2C\_CR1 register. The HSI16 oscillator must be selected as the clock source for I2CCLK in order to allow wakeup from Stop mode.

During Stop mode, the HSI16 is switched off. When a START is detected, the I2C interface switches the HSI16 on, and stretches SCL low until HSI16 is woken up.

HSI16 is then used for the address reception.

In case of an address match, the I2C stretches SCL low during MCU wakeup time. The stretch is released when ADDR flag is cleared by software, and the transfer goes on normally.

If the address does not match, the HSI16 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 HSI16 is not switched on after a START is received.*

*Only an ADDR interrupt can wakeup the MCU. Therefore do not enter Stop mode when the I2C is performing a transfer as a master, or as an addressed slave after the ADDR flag is set. This can be managed by clearing SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.*

**Caution:** The digital filter is not compatible with the wakeup from Stop mode feature. If the DNF bit is not equal to 0, setting the WUPEN bit has no effect.

**Caution:** This feature is available only when the I2C clock source is the HSI16 oscillator.

**Caution:** Clock stretching must be enabled (NOSTRETCH=0) to ensure proper operation of the wakeup from Stop mode feature.

**Caution:** If wakeup from Stop mode is disabled (WUPEN=0), the I2C peripheral must be disabled before entering Stop mode (PE=0).

### 43.4.15 Error conditions

The following are the error conditions which may cause communication to fail.

#### Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of 9 SCL clock pulses. A START or a STOP condition is detected when a SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C is involved in the transfer as master or addressed slave (i.e not during the address phase in slave mode).

In case of a misplaced START or RESTART detection in slave mode, the I2C enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag is set in the I2C\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

### Arbitration lost (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

- In master mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the master switches automatically to slave mode.
- In slave mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped, and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag is set in the I2C\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

### Overrun/underrun error (OVR)

An overrun or underrun error is detected in slave mode when NOSTRETCH=1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF=1 and the first data byte should be sent. The content of the I2C\_TXDR register is sent if TXE=0, 0xFF if not.
  - When a new byte should be sent and the I2C\_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag is set in the I2C\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

### Packet Error Checking Error (PECERR)

This section is relevant only when the SMBus feature is supported. Refer to [Section 43.3: 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 43.3: I2C implementation](#).

A timeout error occurs for any of these conditions:

- TIDLE=0 and SCL remained low for the time defined in the TIMEOUTA[11:0] bits: this is used to detect a SMBus timeout.
- TIDLE=1 and both SDA and SCL remained high for the time defined in the TIMEOUTA [11:0] bits: this is used to detect a bus idle condition.
- Master cumulative clock low extend time reached the time defined in the TIMEOUTB[11:0] bits (SMBus  $t_{\text{LOW:MEXT}}$  parameter)
- Slave cumulative clock low extend time reached the time defined in TIMEOUTB[11:0] bits (SMBus  $t_{\text{LOW:SEXT}}$  parameter)

When a timeout violation is detected in master mode, a STOP condition is automatically sent.

When a timeout violation is detected in slave mode, SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

### Alert (ALERT)

This section is relevant only when the SMBus feature is supported. Refer to [Section 43.3: 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.

## 43.4.16 DMA requests

### Transmission using DMA

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the I2C\_CR1 register. Data is loaded from an SRAM area configured using the DMA peripheral (see [Section 11: Direct memory access controller \(DMA\) on page 348](#)) to the I2C\_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In master mode: the initialization, the slave address, direction, number of bytes and START bit are programmed by software (the transmitted slave address cannot be transferred with DMA). When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to [Master transmitter on page 1471](#).
- In slave mode:
  - With NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
  - With NOSTRETCH=1, the DMA must be initialized before the address match event.
- For instances supporting SMBus: the PEC transfer is managed with NBYTES counter. Refer to [SMBus Slave transmitter on page 1486](#) and [SMBus Master transmitter on page 1489](#).

*Note:* If DMA is used for transmission, the TXIE bit does not need to be enabled.

### Reception using DMA

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the I2C\_CR1 register. Data is loaded from the I2C\_RXDR register to an SRAM area configured using the DMA peripheral (refer to [Section 11: Direct memory access controller \(DMA\) on page 348](#)) 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 43.3: I2C implementation](#)): the PEC transfer is managed with the NBYTES counter. Refer to [SMBus Slave receiver on page 1487](#) and [SMBus Master receiver on page 1491](#).

*Note:* If DMA is used for reception, the RXIE bit does not need to be enabled.

### 43.4.17 Debug mode

When the microcontroller enters debug mode (core halted), the SMBus timeout either continues to work normally or stops, depending on the DBG\_I2Cx\_SMBUS\_TIMEOUT configuration bits in the DBG module.

## 43.5 I2C low-power modes

**Table 261. Effect of low-power modes on the I2C**

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Low-power run	No effect.
Low-power sleep	No effect. I2C interrupts cause the device to exit the Low-power sleep mode.
Stop 0 / Stop 1	The I2C registers content is kept. If WUPEN = 1 and I2C is clocked by HSI16: the address recognition is functional. The I2C address match condition causes the device to exit the Stop 0 and Stop 1 modes. If WUPEN=0: the I2C must be disabled before entering Stop mode.
Stop 2	The I2C registers content is kept. The I2C1, I2C2 and I2C4 must be disabled before entering Stop 2. If WUPEN=1 and I2C3 is clocked by HSI16: the I2C3 address recognition is functional. The I2C3 address match condition causes the device to exit the Stop mode 2. If WUPEN=0: the I2C3 must be disabled before entering Stop 2 mode.
Standby	The I2C peripheral is powered down and must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	

## 43.6 I2C interrupts

The table below gives the list of I2C interrupt requests.

**Table 262. I2C Interrupt requests**

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
Receive buffer not empty	RXNE	Read I2C_RXDR register	RXIE
Transmit buffer interrupt status	TXIS	Write I2C_TXDR register	TXIE

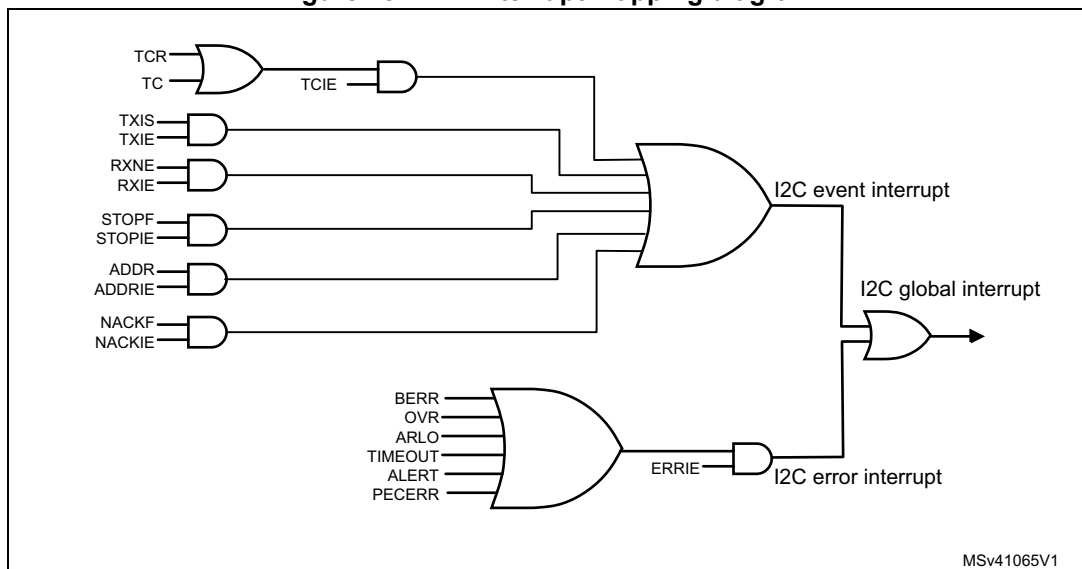
**Table 262. I2C Interrupt requests (continued)**

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
Stop detection interrupt flag	STOPF	Write STOPCF=1	STOPIE
Transfer Complete Reload	TCR	Write I2C_CR2 with NBYTES[7:0] ≠ 0	TCIE
Transfer complete	TC	Write START=1 or STOP=1	
Address matched	ADDR	Write ADDRCF=1	ADDRIE
NACK reception	NACKF	Write NACKCF=1	NACKIE
Bus error	BERR	Write BERRCF=1	ERRIE
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	

Depending on the product implementation, all these interrupts events can either share the same interrupt vector (I2C global interrupt), or be grouped into 2 interrupt vectors (I2C event interrupt and I2C error interrupt). Refer to [Table 69: STM32L4Rxxx and STM32L4Sxxx vector table](#) for details.

The I2C wakeup event is connected to the EXTI controller (refer to [Section 13: Extended interrupts and events controller \(EXTI\)](#)).

**Figure 451. I2C interrupt mapping diagram**



## 43.7 I2C registers

Refer to [Section 1.2 on page 75](#) for a list of abbreviations used in register descriptions.

The peripheral registers are accessed by words (32-bit).

### 43.7.1 Control register 1 (I2C\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to  $2 \times \text{PCLK1} + 6 \times \text{I2CCLK}$ .

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERT EN	SMBDEN	SMBHEN	GCEN	WUPE N	NOSTR ETCH	SBC
								r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMA EN	TXDMA EN	Res.	ANF OFF	DNF				ERRIE	TCIE	STOP IE	NACK IE	ADDR IE	RXIE	TXIE	PE
r/w	r/w		r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **PECEN**: PEC enable

- 0: PEC calculation disabled
- 1: PEC calculation enabled

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 43.3: I2C implementation](#).*

Bit 22 **ALERTEN**: SMBus alert enable

**Device mode (SMBHEN=0):**

- 0: Releases SMBA pin high and Alert Response Address Header disabled: 0001100x followed by NACK.
- 1: Drives SMBA pin low and Alert Response Address Header enables: 0001100x followed by ACK.

**Host mode (SMBHEN=1):**

- 0: SMBus Alert pin (SMBA) not supported.
- 1: SMBus Alert pin (SMBA) supported.

*Note: When ALERTEN=0, the SMBA pin can be used as a standard GPIO.*

*If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 43.3: 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 43.3: 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 43.3: I2C implementation](#).*
- Bit 19 **GCEN**: General call enable  
 0: General call disabled. Address 0b00000000 is NACKed.  
 1: General call enabled. Address 0b00000000 is ACKed.
- Bit 18 **WUPEN**: Wakeup from Stop mode enable  
 0: Wakeup from Stop mode disable.  
 1: Wakeup from Stop mode enable.  
*Note: If the Wakeup from Stop mode feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 43.3: I2C implementation](#).*  
*Note: WUPEN can be set only when DNF = '0000'*
- Bit 17 **NOSTRETCH**: Clock stretching disable  
 This bit is used to disable clock stretching in slave mode. It must be kept cleared in master mode.  
 0: Clock stretching enabled  
 1: Clock stretching disabled  
*Note: This bit can only be programmed when the I2C is disabled (PE = 0).*
- Bit 16 **SBC**: Slave byte control  
 This bit is used to enable hardware byte control in slave mode.  
 0: Slave byte control disabled  
 1: Slave byte control enabled
- Bit 15 **RXDMAEN**: DMA reception requests enable  
 0: DMA mode disabled for reception  
 1: DMA mode enabled for reception
- Bit 14 **TXDMAEN**: DMA transmission requests enable  
 0: DMA mode disabled for transmission  
 1: DMA mode enabled for transmission
- Bit 13 Reserved, must be kept at reset value.
- Bit 12 **ANFOFF**: Analog noise filter OFF  
 0: Analog noise filter enabled  
 1: Analog noise filter disabled  
*Note: This bit can only be programmed when the I2C is disabled (PE = 0).*
- Bits 11:8 **DNF[3:0]**: Digital noise filter  
 These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter will filter spikes with a length of up to  $DNF[3:0] * t_{I2CCLK}$   
 0000: Digital filter disabled  
 0001: Digital filter enabled and filtering capability up to  $1 t_{I2CCLK}$   
 ...  
 1111: digital filter enabled and filtering capability up to  $15 t_{I2CCLK}$   
*Note: If the analog filter is also enabled, the digital filter is added to the analog filter. This filter can only be programmed when the I2C is disabled (PE = 0).*

Bit 7 **ERRIE**: Error interrupts enable

0: Error detection interrupts disabled

1: Error detection interrupts enabled

*Note: Any of these errors generate an interrupt:*

*Arbitration Loss (ARLO)*

*Bus Error detection (BERR)*

*Overrun/Underrun (OVR)*

*Timeout detection (TIMEOUT)*

*PEC error detection (PECERR)*

*Alert pin event detection (ALERT)*

Bit 6 **TCIE**: Transfer Complete interrupt enable

0: Transfer Complete interrupt disabled

1: Transfer Complete interrupt enabled

*Note: Any of these events will generate an interrupt:*

*Transfer Complete (TC)*

*Transfer Complete Reload (TCR)*

Bit 5 **STOPIE**: Stop detection Interrupt enable

0: Stop detection (STOPF) interrupt disabled

1: Stop detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received Interrupt enable

0: Not acknowledge (NACKF) received interrupts disabled

1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match Interrupt enable (slave only)

0: Address match (ADDR) interrupts disabled

1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX Interrupt enable

0: Receive (RXNE) interrupt disabled

1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX Interrupt enable

0: Transmit (TXIS) interrupt disabled

1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable

0: Peripheral disable

1: Peripheral enable

*Note: When PE=0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least 3 APB clock cycles.*

### 43.7.2 Control register 2 (I2C\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PEC BYTE	AUTO END	RE LOAD	NBYTES[7:0]							
					rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD 10R	ADD10	RD_WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **PECBYTE**: Packet error checking byte

This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE=0.

0: No PEC transfer.

1: PEC transmission/reception is requested

*Note: Writing '0' to this bit has no effect.*

*This bit has no effect when RELOAD is set.*

*This bit has no effect is 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 43.3: I2C implementation](#).*

Bit 25 **AUTOEND**: Automatic end mode (master mode)

This bit is set and cleared by software.

0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.

1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

*Note: This bit has no effect in slave mode or when the RELOAD bit is set.*

Bit 24 **RELOAD**: NBYTES reload mode

This bit is set and cleared by software.

0: The transfer is completed after the NBYTES data transfer (STOP or RESTART will follow).

1: The transfer is not completed after the NBYTES data transfer (NBYTES will be reloaded).

TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in slave mode with SBC=0.

*Note: Changing these bits when the START bit is set is not allowed.*

**Bit 15 NACK:** NACK generation (slave mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE=0.

- 0: an ACK is sent after current received byte.
- 1: a NACK is sent after current received byte.

*Note: Writing '0' to this bit has no effect.*

*This bit is used in slave mode only: in master receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.*

*When an overrun occurs in slave receiver NOSTRETCH mode, a NACK is automatically generated whatever the NACK bit value.*

*When hardware PEC checking is enabled (PECBYTE=1), the PEC acknowledge value does not depend on the NACK value.*

**Bit 14 STOP:** Stop generation (master mode)

The bit is set by software, cleared by hardware when a STOP condition is detected, or when PE = 0.

**In Master Mode:**

- 0: No Stop generation.
- 1: Stop generation after current byte transfer.

*Note: Writing '0' to this bit has no effect.*

**Bit 13 START:** Start generation

This bit is set by software, and cleared by hardware after the Start followed by the address sequence is sent, by an arbitration loss, by a timeout error detection, or when PE = 0. It can also be cleared by software by writing '1' to the ADDRCONF bit in the I2C\_ICR register.

- 0: No Start generation.
- 1: Restart/Start generation:

- If the I2C is already in master mode with AUTOEND = 0, setting this bit generates a Repeated Start condition when RELOAD=0, after the end of the NBYTES transfer.
- Otherwise setting this bit will generate a START condition once the bus is free.

*Note: Writing '0' to this bit has no effect.*

*The START bit can be set even if the bus is BUSY or I2C is in slave mode.*

*This bit has no effect when RELOAD is set. In 10-bit addressing mode, if a NACK is received on the first part of the address, the START bit is not cleared by hardware and the master will resend the address sequence, unless the START bit is cleared by software*

**Bit 12 HEAD10R:** 10-bit address header only read direction (master receiver mode)

- 0: The master sends the complete 10 bit slave address read sequence: Start + 2 bytes 10bit address in write direction + Restart + 1st 7 bits of the 10 bit address in read direction.
- 1: The master only sends the 1st 7 bits of the 10 bit address, followed by Read direction.

*Note: Changing this bit when the START bit is set is not allowed.*

**Bit 11 ADD10:** 10-bit addressing mode (master mode)

- 0: The master operates in 7-bit addressing mode,
- 1: The master operates in 10-bit addressing mode

*Note: Changing this bit when the START bit is set is not allowed.*

**Bit 10 RD\_WRN:** Transfer direction (master mode)

- 0: Master requests a write transfer.
- 1: Master requests a read transfer.

*Note: Changing this bit when the START bit is set is not allowed.*

Bits 9:8 **SADD[9:8]**: Slave address bit 9:8 (master mode)

**In 7-bit addressing mode (ADD10 = 0):**

These bits are don't care

**In 10-bit addressing mode (ADD10 = 1):**

These bits should be written with bits 9:8 of the slave address to be sent

*Note: Changing these bits when the START bit is set is not allowed.*

Bits 7:1 **SADD[7:1]**: Slave address bit 7:1 (master mode)

**In 7-bit addressing mode (ADD10 = 0):**

These bits should be written with the 7-bit slave address to be sent

**In 10-bit addressing mode (ADD10 = 1):**

These bits should be written with bits 7:1 of the slave address to be sent.

*Note: Changing these bits when the START bit is set is not allowed.*

Bit 0 **SADD0**: Slave address bit 0 (master mode)

**In 7-bit addressing mode (ADD10 = 0):**

This bit is don't care

**In 10-bit addressing mode (ADD10 = 1):**

This bit should be written with bit 0 of the slave address to be sent

*Note: Changing these bits when the START bit is set is not allowed.*

### 43.7.3 Own address 1 register (I2C\_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 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[9:8]		OA1[7:1]							OA1[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 9:8 **OA1[9:8]**: Interface address

- 7-bit addressing mode: do not care
- 10-bit addressing mode: bits 9:8 of address

*Note: These bits can be written only when OA1EN=0.*

Bits 7:1 **OA1[7:1]**: Interface address

- 7-bit addressing mode: 7-bit address
- 10-bit addressing mode: bits 7:1 of 10-bit address

*Note: These bits can be written only when OA1EN=0.*

Bit 0 **OA1[0]**: Interface address

- 7-bit addressing mode: do not care
- 10-bit addressing mode: bit 0 of address

*Note: This bit can be written only when OA1EN=0.*

### 43.7.4 Own address 2 register (I2C\_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 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
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA2EN**: Own Address 2 enable

- 0: Own address 2 disabled. The received 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.

### 43.7.5 Timing register (I2C\_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]			
rw	rw	rw	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 on page 1452](#)) and for SCL high and low level counters (refer to [I2C master initialization on page 1467](#)).

$$t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$$

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:  $SDADEL$  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:  $SCLH$  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:  $SCLL$  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.*



### 43.7.6 Timeout register (I2C\_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to  $2 \times PCLK1 + 6 \times I2CCLK$ .

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]											
r/w				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
r/w			r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bit 31 **TEXTEN**: Extended clock timeout enable

0: Extended clock timeout detection is disabled

1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than  $t_{LOW:EXT}$  is done by the I2C interface, a timeout error is detected (TIMEOUT=1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B

This field is used to configure the cumulative clock extension timeout:

In master mode, the master cumulative clock low extend time ( $t_{LOW:MEXT}$ ) is detected

In slave mode, the slave cumulative clock low extend time ( $t_{LOW:SEXT}$ ) is detected

$$t_{LOW:EXT} = (TIMEOUTB + 1) \times 2048 \times t_{I2CCLK}$$

Note: These bits can be written only when TEXTEN=0.

Bit 15 **TIMOUTEN**: Clock timeout enable

0: SCL timeout detection is disabled

1: SCL timeout detection is enabled: when SCL is low for more than  $t_{TIMEOUT}$  (TIDLE=0) or high for more than  $t_{IDLE}$  (TIDLE=1), a timeout error is detected (TIMEOUT=1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection

0: TIMEOUTA is used to detect SCL low timeout

1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)

Note: This bit can be written only when TIMOUTEN=0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus Timeout A

This field is used to configure:

– The SCL low timeout condition  $t_{TIMEOUT}$  when TIDLE=0

$$t_{TIMEOUT} = (TIMEOUTA + 1) \times 2048 \times t_{I2CCLK}$$

– The bus idle condition (both SCL and SDA high) when TIDLE=1

$$t_{IDLE} = (TIMEOUTA + 1) \times 4 \times t_{I2CCLK}$$

Note: These bits can be written only when TIMOUTEN=0.

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to "0x00000000". Refer to [Section 43.3: I2C implementation](#).

### 43.7.7 Interrupt and status register (I2C\_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r	r	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 43.3: I2C implementation.*

Bit 12 **TIMEOUT**: Timeout or t<sub>LOW</sub> detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

*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 43.3: 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 43.3: I2C implementation](#).*

**Bit 10 OVR:** Overrun/Underrun (slave mode)

This flag is set by hardware in slave mode with NOSTRETCH=1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

*Note:* This bit is cleared by hardware when PE=0.

**Bit 9 ARLO:** Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

*Note:* This bit is cleared by hardware when PE=0.

**Bit 8 BERR:** Bus error

This flag is set by hardware when a misplaced Start or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in slave mode. It is cleared by software by setting BERRCF bit.

*Note:* This bit is cleared by hardware when PE=0.

**Bit 7 TCR:** Transfer Complete Reload

This flag is set by hardware when RELOAD=1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

*Note:* This bit is cleared by hardware when PE=0.

*This flag is only for master mode, or for slave mode when the SBC bit is set.*

**Bit 6 TC:** Transfer Complete (master mode)

This flag is set by hardware when RELOAD=0, AUTOEND=0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

*Note:* This bit is cleared by hardware when PE=0.

**Bit 5 STOPF:** Stop detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

- either as a master, provided that the STOP condition is generated by the peripheral.
- or as a slave, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

*Note:* This bit is cleared by hardware when PE=0.

**Bit 4 NACKF:** Not Acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

*Note:* This bit is cleared by hardware when PE=0.

**Bit 3 ADDR:** Address matched (slave mode)

This bit is set by hardware as soon as the received slave address matched with one of the enabled slave addresses. It is cleared by software by setting ADDRCF bit.

*Note:* This bit is cleared by hardware when PE=0.

- Bit 2 **RXNE**: Receive data register not empty (receivers)  
 This bit is set by hardware when the received data is copied into the I2C\_RXDR register, and is ready to be read. It is cleared when I2C\_RXDR is read.  
*Note: This bit is cleared by hardware when PE=0.*
- Bit 1 **TXIS**: Transmit interrupt status (transmitters)  
 This bit is set by hardware when the I2C\_TXDR register is empty and the data to be transmitted must be written in the I2C\_TXDR register. It is cleared when the next data to be sent is written in the I2C\_TXDR register.  
 This bit can be written to '1' by software when NOSTRETCH=1 only, in order to generate a TXIS event (interrupt if TXIE=1 or DMA request if TXDMAEN=1).  
*Note: This bit is cleared by hardware when PE=0.*
- Bit 0 **TXE**: Transmit data register empty (transmitters)  
 This bit is set by hardware when the I2C\_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C\_TXDR register.  
 This bit can be written to '1' by software in order to flush the transmit data register I2C\_TXDR.  
*Note: This bit is set by hardware when PE=0.*

### 43.7.8 Interrupt clear register (I2C\_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERT CF	TIM OUTCF	PECCF	OVRCF	ARLO CF	BERR CF	Res.	Res.	STOP CF	NACK CF	ADDR CF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

Bits 31:14 Reserved, must be kept at reset value.

- Bit 13 **ALERTCF**: Alert flag clear  
 Writing 1 to this bit clears the ALERT flag in the I2C\_ISR register.  
*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 43.3: 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 43.3: 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 43.3: 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.

### 43.7.9 PEC register (I2C\_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]									
								r	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 43.3: I2C implementation](#).

### 43.7.10 Receive data register (I2C\_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
								r	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.

### 43.7.11 Transmit data register (I2C\_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXDATA[7:0]** 8-bit transmit data

Data byte to be transmitted to the I<sup>2</sup>C bus.

*Note: These bits can be written only when TXE=1.*

### 43.7.12 I2C register map

The table below provides the I2C register map and reset values.

**Table 263. I2C register map and reset values**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x0	I2C_CR1	Res	Res	Res	Res	Res	Res	Res	Res	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res	ANFOFF	DNF[3:0]			ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE		
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x4	I2C_CR2	Res	Res	Res	Res	Res	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]							NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]											
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x8	I2C_OAR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OA1EN	Res	Res	Res	Res	OA1MODE	OA1[9:0]										
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0xC	I2C_OAR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OA2EN	Res	Res	Res	Res	Res	OA2MSK[2:0]	OA2[7:1]					Res				
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x10	I2C_TIMINGR	PRESC[3:0]			Res	Res	Res	Res	Res	SCLDEL[3:0]	SDADEL[3:0]	SCLH[7:0]					SCLL[7:0]																	
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x14	I2C_TIMEOUTR	TEXTEN	Res	Res	Res	TIMEOUTB[11:0]										TIMEOUTEN	Res	TIDLE	TIMEOUTA[11:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x18	I2C_ISR	Res	Res	Res	Res	Res	Res	Res	Res	ADDCODE[6:0]							DIR	BUSY	Res	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDRF	RXNE	TXIS	TXE	
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
0x1C	I2C_ICR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ALERTCF	TIMEOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res	Res	STOPCF	NACKCF	ADDRCF	Res	Res	Res
	Reset value																				0	0	0	0	0	0			0	0	0			
0x20	I2C_PECR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
	Reset value																											0	0	0	0	0	0	0
0x24	I2C_RXDR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
	Reset value																											0	0	0	0	0	0	0

Table 263. 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.



## 44 Universal synchronous asynchronous receiver transmitter (USART)

This section describes the universal synchronous asynchronous receiver transmitter (USART).

### 44.1 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.

## 44.2 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

### 44.3 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

### 44.4 USART implementation

[Table 264](#) describes USART implementation on STM32L4Rxxx and STM32L4Sxxx devices.

**Table 264. USART/LPUART features**

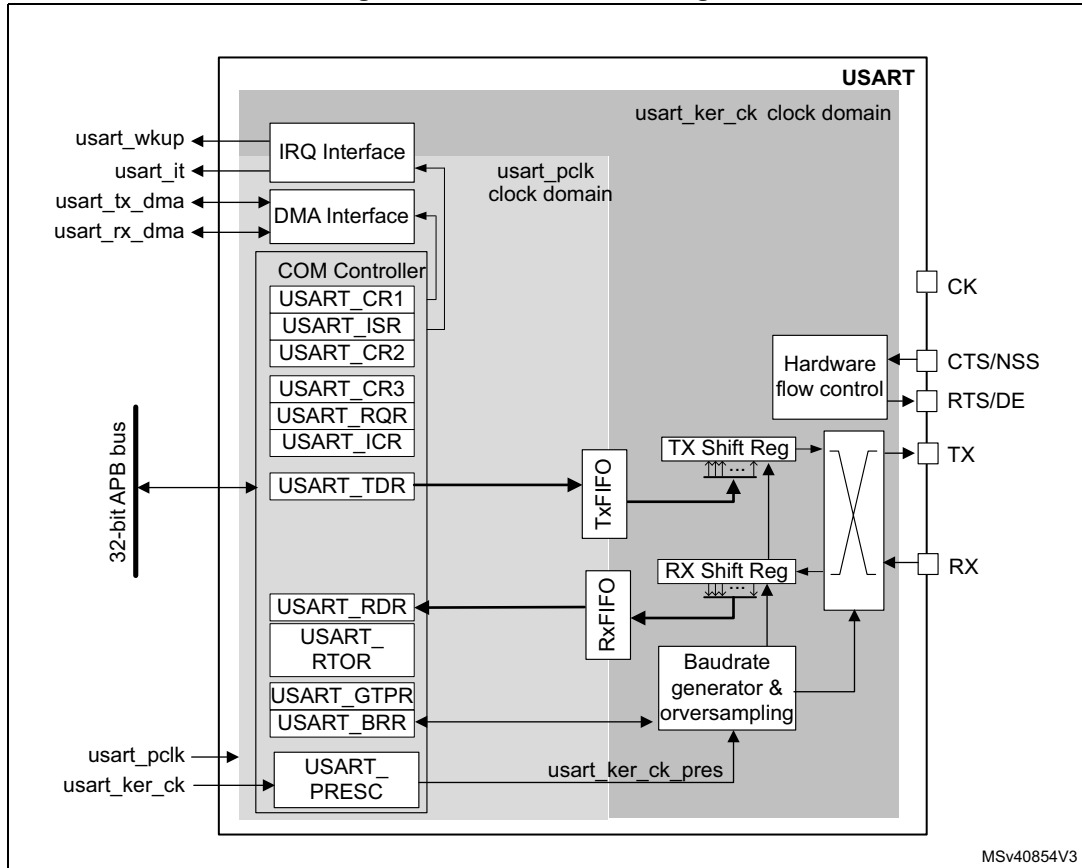
USART modes/features <sup>(1)</sup>	USART1/2/3	UART4/5	LPUART
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (Master/Slave)	X	-	-
Smartcard mode	X	-	-
Single-wire Half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual clock domain and wakeup from low-power mode	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver Enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size	8		

1. X = supported.

## 44.5 USART functional description

### 44.5.1 USART block diagram

Figure 452. USART block diagram



The simplified block diagram given in [Figure 452](#) 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 SCLK 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.

## 44.5.2 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.

### 44.5.3 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 453](#)):

- 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.

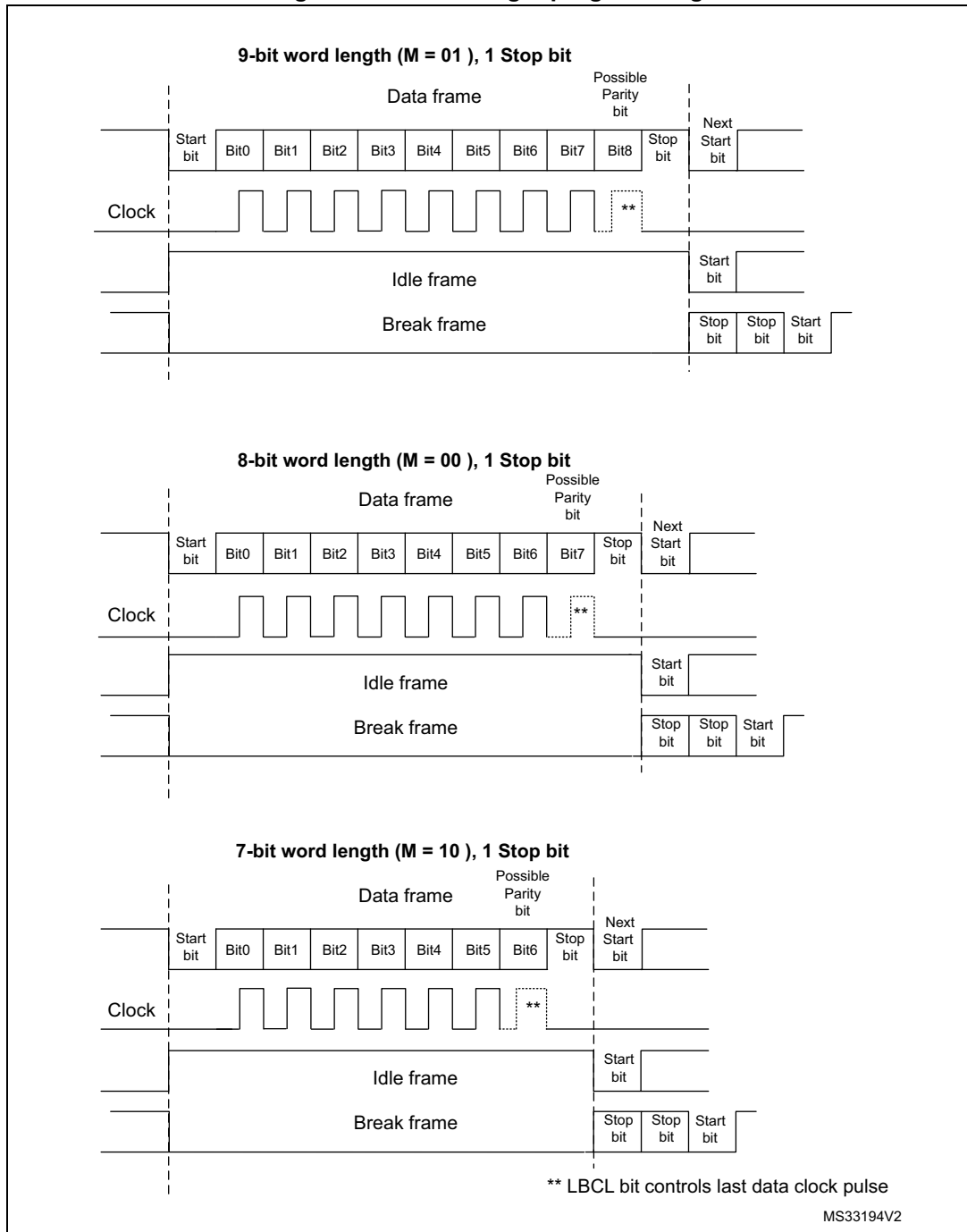
An **Idle character** is interpreted as an entire frame of "1"s (the number of "1"s includes the number of stop bits).

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 453. Word length programming



#### 44.5.4 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 Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.

In this case, the RXFT flag is set in the USART\_ISR register. This means that RXFTCFG data have been received: 1 data in USART\_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to '101', the RXFT flag will be 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 will not set the overrun flag.

- The Tx interrupt is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.

#### 44.5.5 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 SCLK 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**

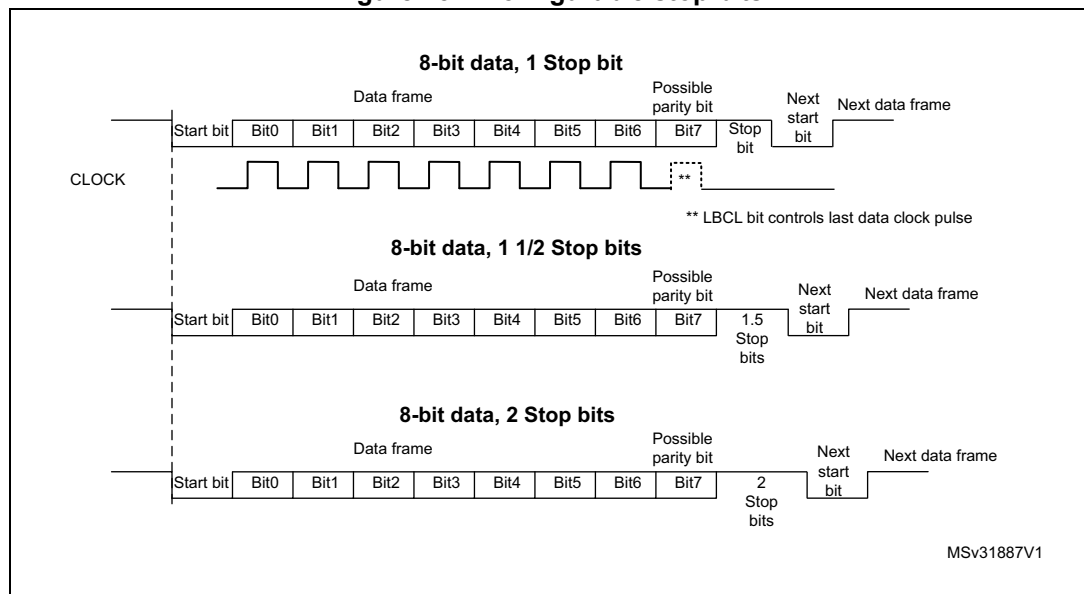
The number of stop bits to be transmitted with every character can be programmed in USART\_CR2, bits 13,12.

- **1 stop bit:** This is the default value of number of stop bits.
- **2 stop bits:** This will be supported by normal USART, Single-wire and Modem modes.
- **1.5 stop bits:** To be used in Smartcard mode.

An idle frame transmission will include the stop bits.

A break transmission will be 10 low bits (when M[1:0] = '00') or 11 low bits (when M[1:0] = '01') or 9 low bits (when M[1:0] = '10') followed by 2 stop bits (see [Figure 454](#)). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

**Figure 454. 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 44.5.10: 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.

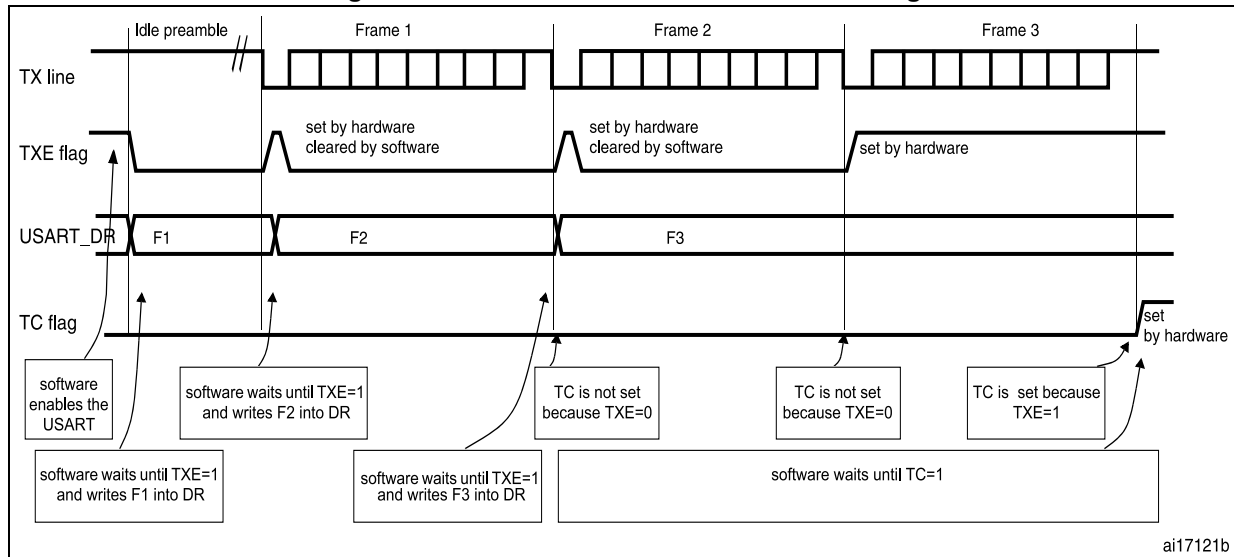
When the TXFIFO is not full, the TXFNF flag stays at '1' even after a write operation to USART\_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

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 microcontroller to enter the low-power mode (see [Figure 455: TC/TXE behavior when transmitting](#)).

Figure 455. 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 453).

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.

**44.5.6 USART receiver**

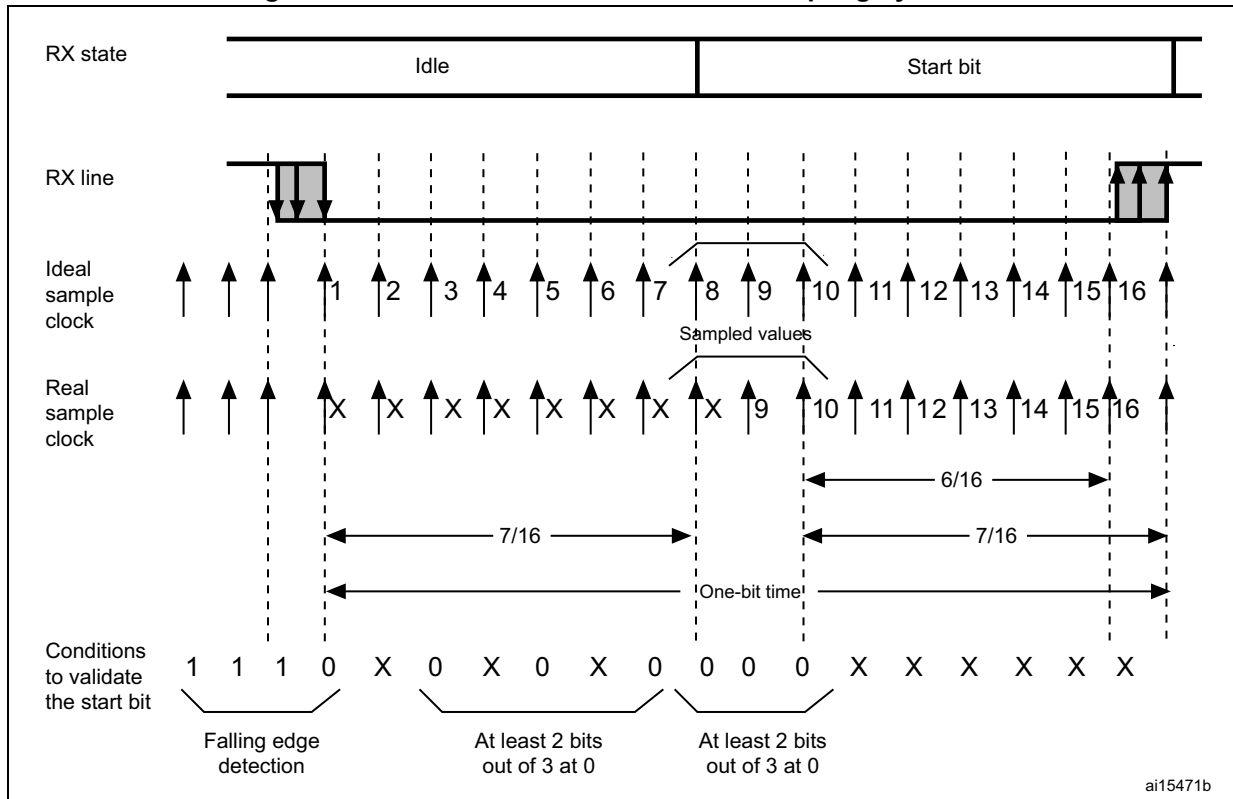
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART\_CR1 register.

**Start bit detection**

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.

Figure 456. 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 44.5.10: 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 RXNE flag is set after every byte reception.

An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  - the ORE bit is set;
  - the RDR content will not be lost. The previous data is available by reading the USART\_RDR register.
  - the shift register will be 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 will not be lost. It is available by reading the USART\_RDR register.
  - The shift register will be 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 ). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

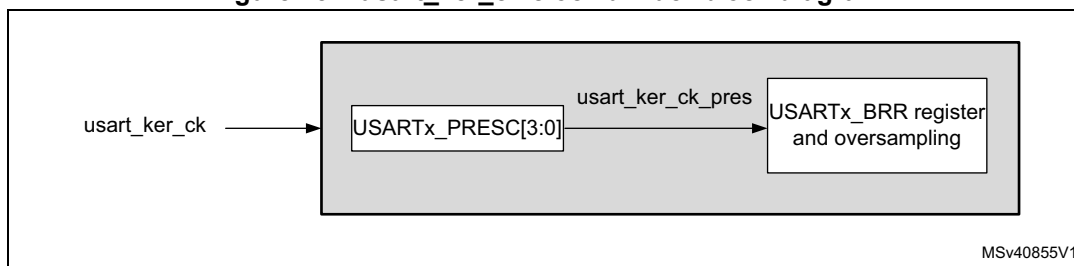
- Possible use of the USART in low-power mode
- Communication speed.

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 ). 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 457. usart\_ker\_ck clock divider block diagram**



Some usart\_ker\_ck sources allow 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 allow 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 allows 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 458](#) and [Figure 459](#)).



Depending on your application:

- select oversampling by 8 (OVER8=1) to achieve higher speed (up to  $\text{usart\_ker\_ck\_pres}/8$ ). In this case the maximum receiver tolerance to clock deviation is reduced (refer to [Section 44.5.8: Tolerance of the USART receiver to clock deviation on page 1535](#))
- select oversampling by 16 (OVER8=0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum  $\text{usart\_ker\_ck\_pres}/16$  (where  $\text{usart\_ker\_ck\_pres}$  is the USART input clock divided by a prescaler).

Programming the ONEBIT bit in the USART\_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.
- A single sample in the center of the received bit

Depending on your application:

- select the three sample majority vote method (ONEBIT=0) when operating in a noisy environment and reject the data when a noise is detected (refer to [Figure 265](#)) 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 44.5.8: Tolerance of the USART receiver to clock deviation on page 1535](#)). In this case the NE bit will never be set.

When noise is detected in a frame:

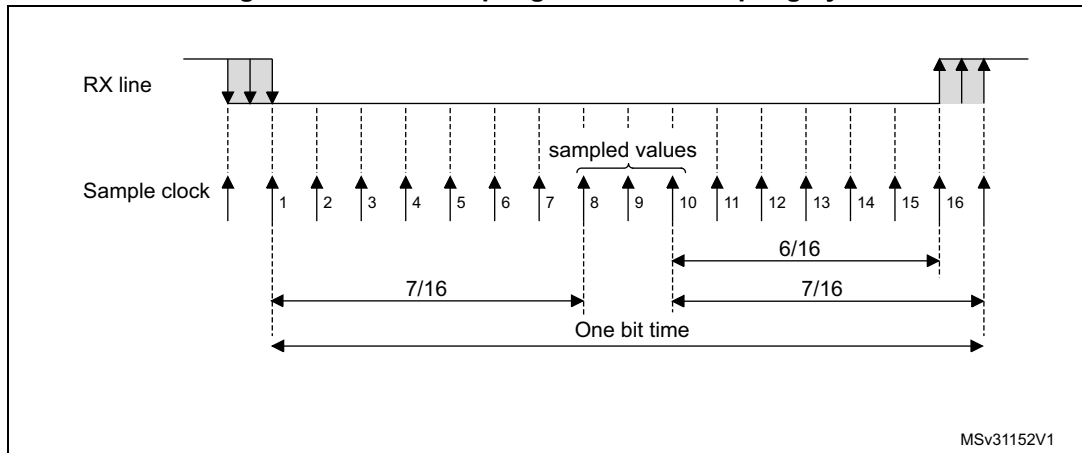
- The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART\_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt will be issued if the EIE bit is set in the USART\_CR3 register.

The NE bit is reset by setting NFCF bit in ICR register.

*Note:* Noise error is not supported in SPI mode.

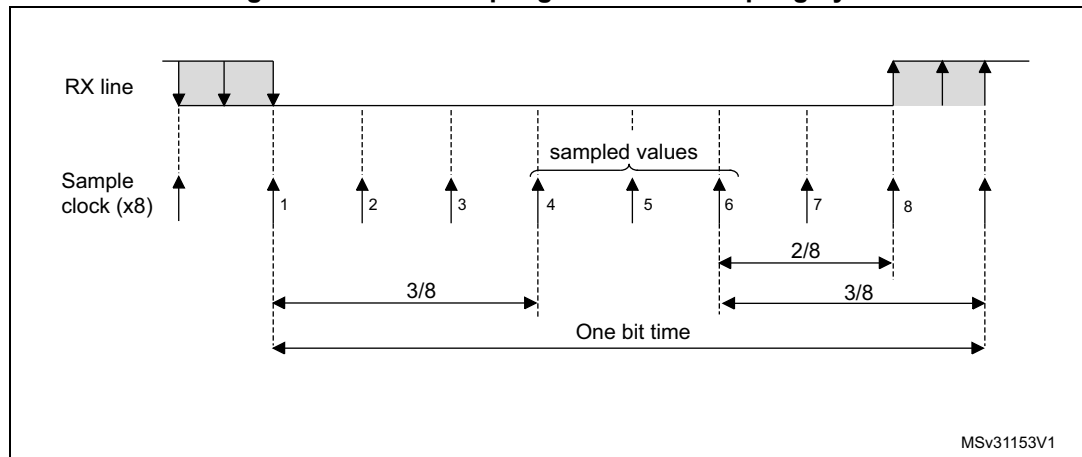
*Oversampling by 8 is not available in the Smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to '0' by hardware.*

Figure 458. Data sampling when oversampling by 16



MSv31152V1

Figure 459. Data sampling when oversampling by 8



MSv31153V1

Table 265. 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 will be 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 44.5.16: USART receiver timeout on page 1548](#) 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.

### 44.5.7 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 = 833d = 0341h
- In case of oversampling by 8:
  - USARTDIV = 2 \* 8 000 000/9600
  - USARTDIV = 1666,66 (1667d = 683h)
  - BRR[3:0] = 3h >>1 = 1h
  - BRR = 0x681

**Example 2**

To obtain 921.6 Kbaud with usart\_ker\_ck\_pres = 48 MHz:

- In case of oversampling by 16:  
 $USARTDIV = 48\,000\,000/921\,600$   
 $BRR = USARTDIV = 52d = 34h$
- In case of oversampling by 8:  
 $USARTDIV = 2 * 48\,000\,000/921\,600$   
 $USARTDIV = 104 (104d = 68h)$   
 $BRR[3:0] = USARTDIV[3:0] \gg 1 = 8h \gg 1 = 4h$   
 $BRR = 0x64$

**44.5.8 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:

$$DWU = \frac{t_{WUUSART}}{11 \times T_{bit}}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUUSART}}{10 \times T_{bit}}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUUSART}}{9 \times T_{bit}}$$

$t_{WUUSART}$  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 266](#), [Table 267](#), 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 266. 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 267. 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 266](#) and [Table 267](#) 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).

### 44.5.9 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 allows the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from  $\text{usart\_ker\_ck\_pres}/65535$  and  $\text{usart\_ker\_ck\_pres}/16$ .
- When oversampling by 8, the baud rate ranges from  $\text{usart\_ker\_ck\_pres}/65535$  and  $\text{usart\_ker\_ck\_pres}/8$ .

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART\_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is

measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are the following:

- **Mode 0:** Any character starting with a bit at '1'.  
In this case the USART measures the duration of the start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.  
In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.
- **Mode 2:** A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).  
In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3:** A 0x55 character frame.  
In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART\_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART\_CR2 register. The USART will then wait for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the 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 will be set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The 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.

### 44.5.10 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.

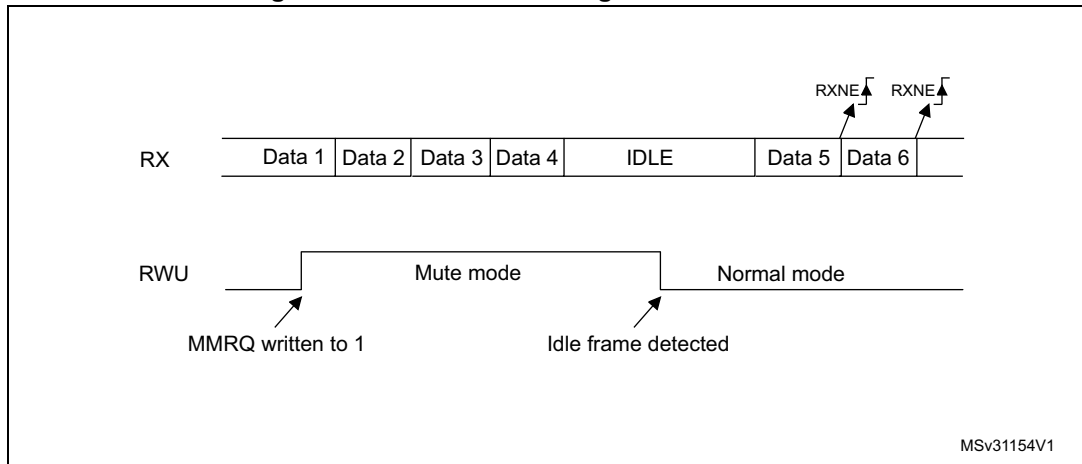
#### Idle line detection (WAKE=0)

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 460](#).



Figure 460. Mute mode using Idle line detection



*Note:* If the MMRQ is set while the IDLE character has already elapsed, Mute mode will not be entered (RWU is not set).  
 If the USART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

**4-bit/7-bit address mark detection (WAKE=1)**

In this mode, bytes are recognized as addresses if their MSB is a '1', otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the 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.

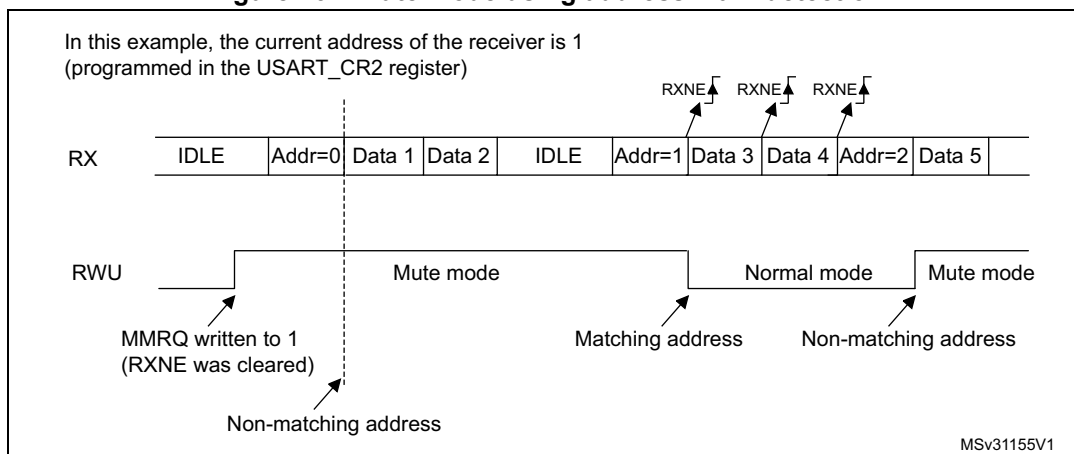
The USART also enters Mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

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 461](#).

Figure 461. Mute mode using address mark detection



### 44.5.11 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.

### 44.5.12 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 268](#).

**Table 268. 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, then the parity bit will be 0 if even parity is selected (PS bit in USART\_CR1 = 0).

#### Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 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 will be 1 if odd parity is selected (PS bit in USART\_CR1 = 1).

#### Parity checking in reception

If the parity check fails, the PE flag is set in the USART\_ISR register and an interrupt is generated if PEIE is set in the USART\_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART\_ICR register.

#### Parity generation in transmission

If the PCE bit is set in USART\_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

### 44.5.13 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to [Section 44.4: USART implementation on page 1517](#).

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 44.5.4](#) 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 2 bits of value '1 are sent to allow the next start detection.

#### LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE=1 in 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 will be the case for any break frame), the receiver stops until the break detection circuit receives either a '1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the [Figure 462: Break detection in LIN mode \(11-bit break length - LBDL bit is set\) on page 1543](#).

Examples of break frames are given on [Figure 463: Break detection in LIN mode vs. Framing error detection on page 1544](#).

**Figure 462. Break detection in LIN mode (11-bit break length - LBDL bit is set)**

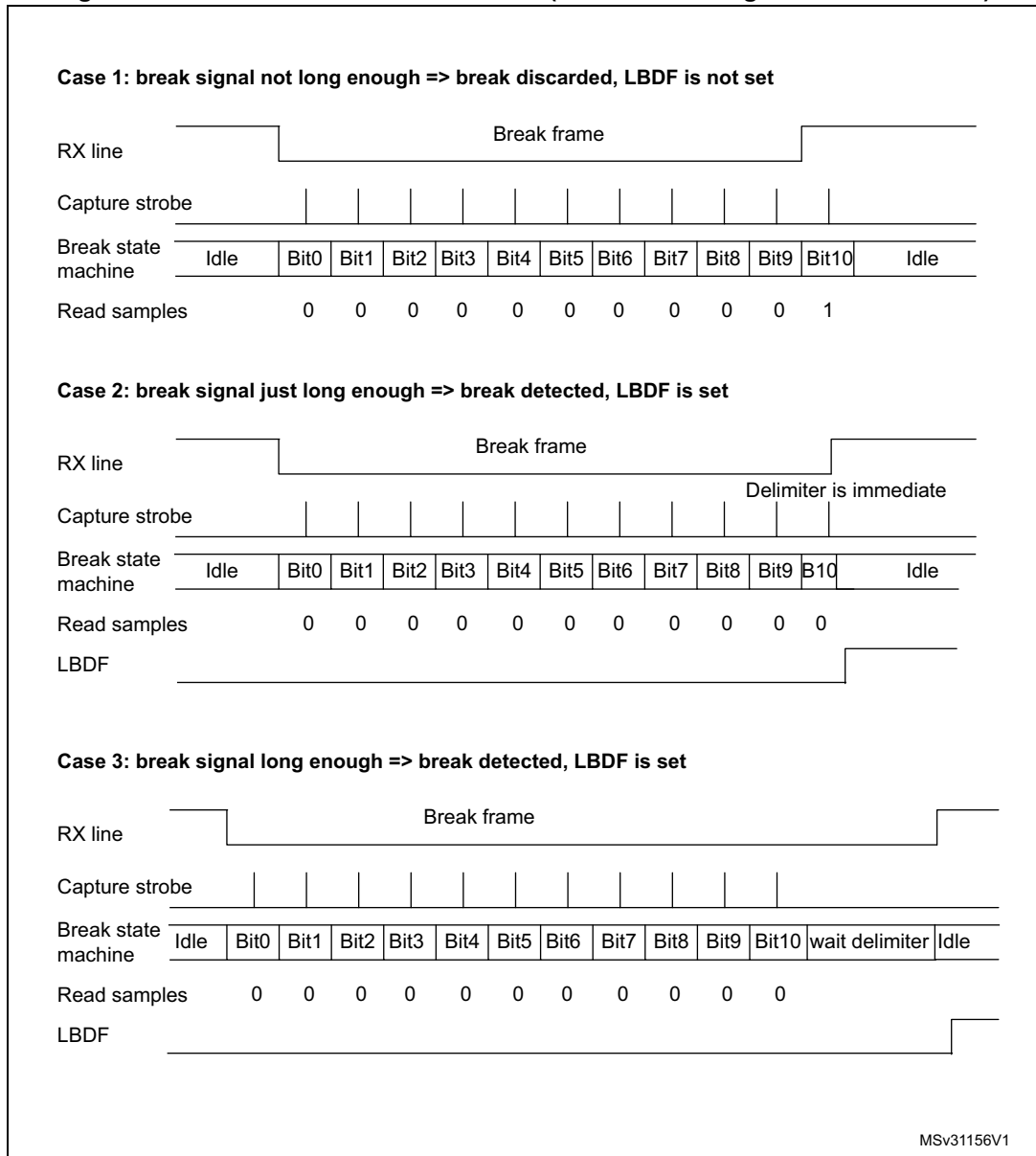
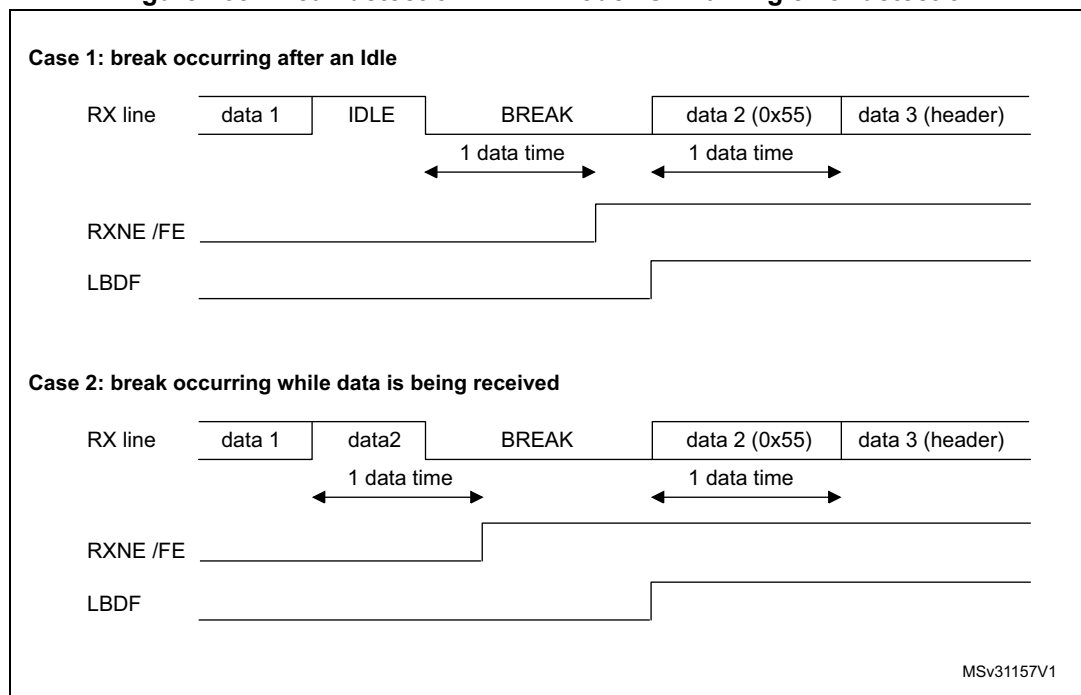


Figure 463. Break detection in LIN mode vs. Framing error detection



#### 44.5.14 USART synchronous mode

##### Master mode

The synchronous master mode is selected by programming the CLKEN bit in the USART\_CR2 register to '1'. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART\_CR2 register,
- SCEN, HDSEL and IREN bits in the USART\_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The SCLK pin is the output of the USART transmitter clock. No clock pulses are sent to the SCLK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART\_CR2 register, clock pulses 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 464](#), [Figure 465](#) and [Figure 466](#)).

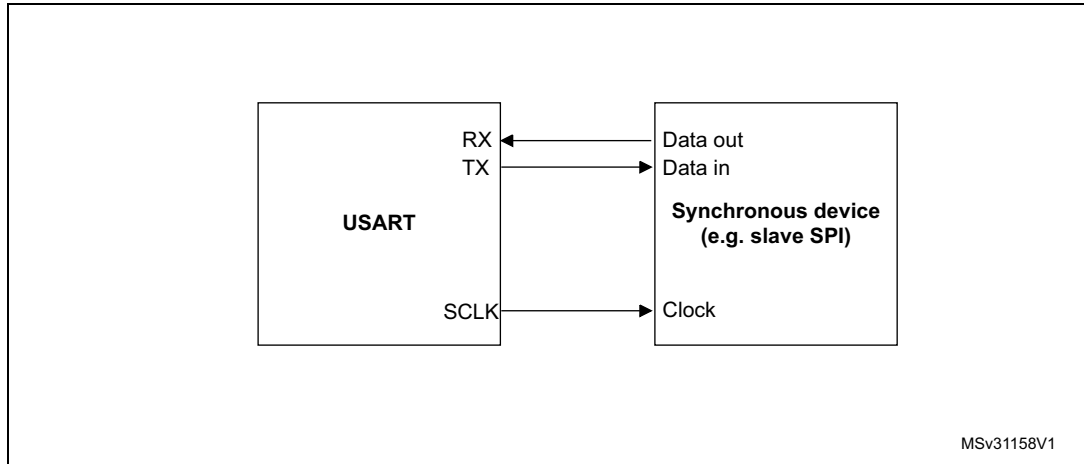
During the Idle state, preamble and send break, the external SCLK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since SCLK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

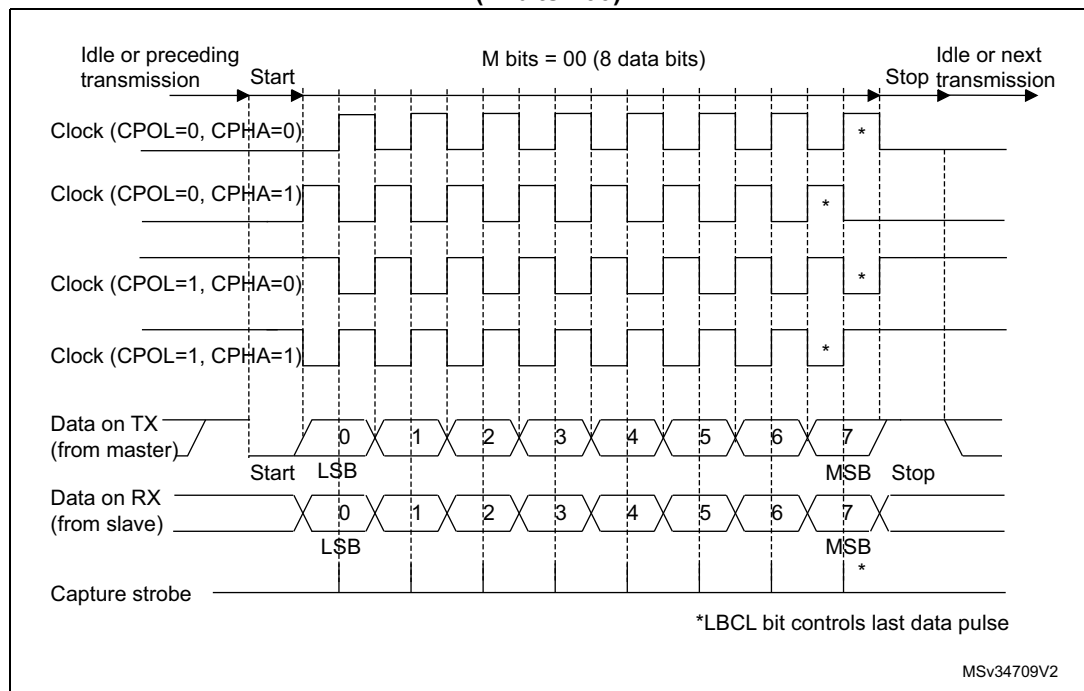
In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on SCLK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

*Note:* In master mode, the SCLK 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.

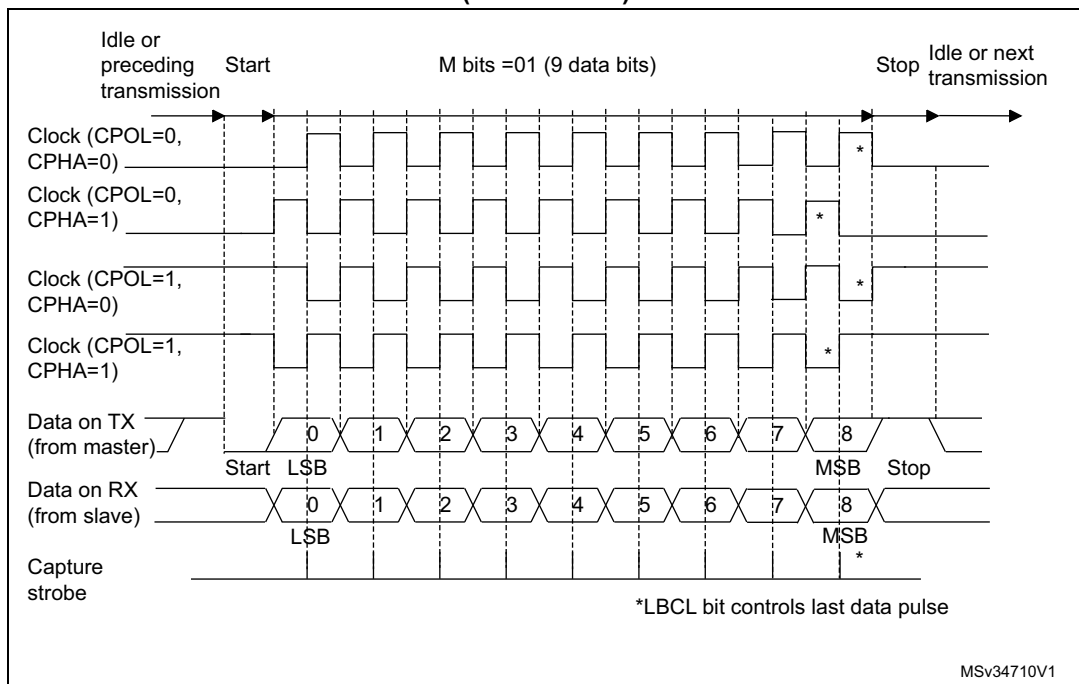
**Figure 464. USART example of synchronous master transmission**



**Figure 465. USART data clock timing diagram in synchronous master mode (M bits =00)**



**Figure 466. 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 SCLK pin is the input of the USART in slave mode.

*Note:* When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (usart\_ker\_ck\_pres) must be greater than 3 times the CK input frequency.

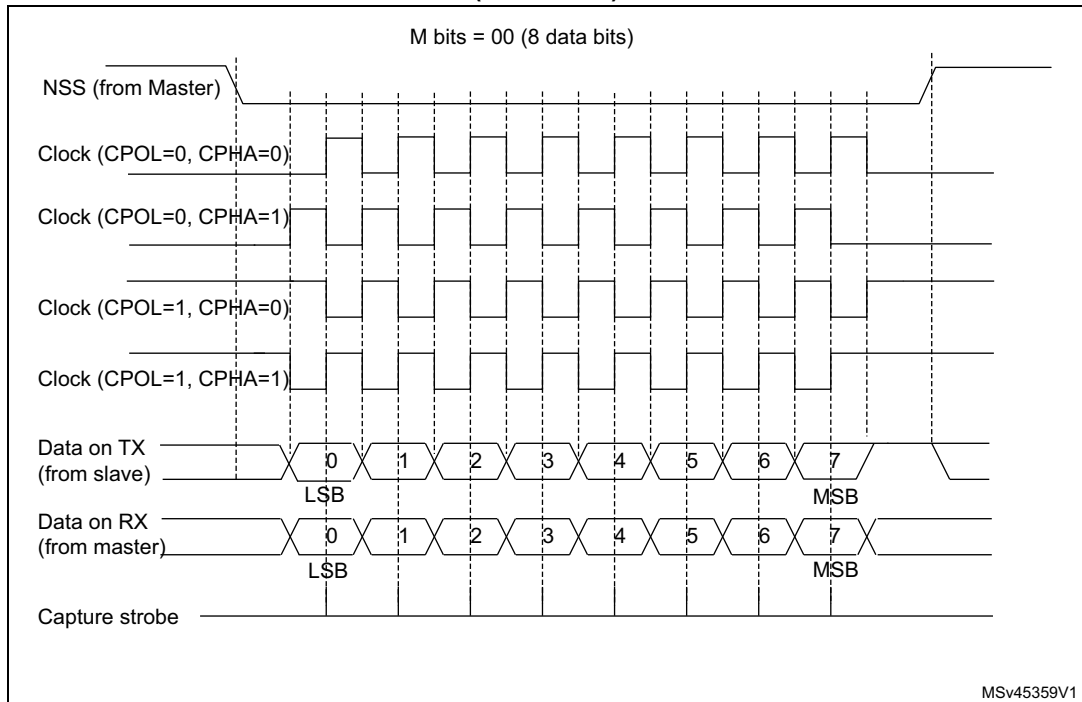
The CPOL bit and the CPHA bit in the USART\_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see [Figure 467](#)).

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 467. 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)  
 SPI slave will always be selected and NSS input pin will be 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 will become 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 will transmit 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 will allow 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 SCLK transmission edge. To avoid this underrun error, the USART\_TDR should be written 3 usart\_ker\_ck cycles before the first SCLK edge.

#### 44.5.15 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.

#### 44.5.16 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 will be generated if RTOIE bit in USART\_CR1 register is set.

### 44.5.17 USART Smartcard mode

This section is relevant only when Smartcard mode is supported. Refer to [Section 44.4: USART implementation on page 1517](#).

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.

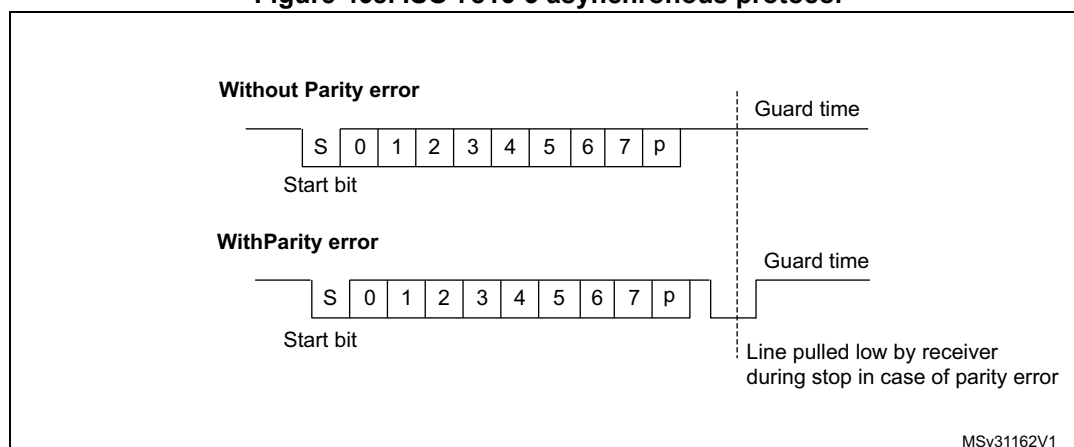
The USART should be configured as:

- 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 468](#) shows examples of what can be seen on the data line with and without parity error.

**Figure 468. ISO 7816-3 asynchronous protocol**



When connected to a Smartcard, the TX output of the USART drives a bidirectional line that is also driven by the Smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the Smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.

The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART\_RQR register.

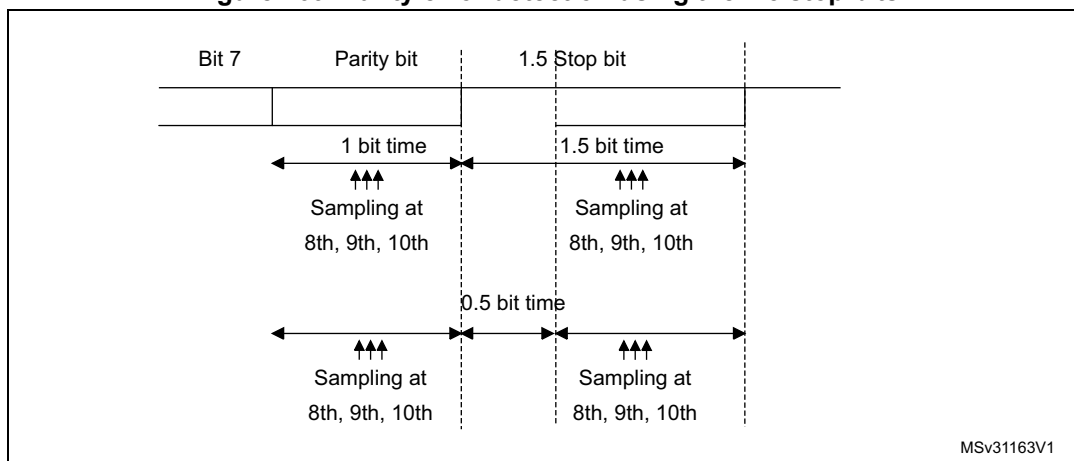
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it again, it must insure the minimum 2 baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the Smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T=1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the Smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn't repeat the character.
- In transmission, the USART inserts the Guard Time (as programmed in the Guard Time register) between two successive characters. As the Guard Time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT (Character Guard Time, as defined by the 7816-3 specification) minus 12 (the duration of one character).
- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the Guard Time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the Guard Time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.
- The de-assertion of TC flag is unaffected by Smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

*Note:* 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 469* 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 469. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the Smartcard through the SCLK output. In Smartcard mode, SCLK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART\_GTPR register. SCLK 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 will be 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 allow 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 will signal 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 will be programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBFF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

*Note:* The error checking code (LRC/CRC) must be computed/verified by software.

### Direct and inverse convention

The Smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=0, DATAINV=0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=1, DATAINV=1.

*Note:* When logical data values are inverted (0=H, 1=L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH => the USART received character will be '03' and the parity will be odd.

Therefore, two methods are available for TS pattern recognition:

#### Method 1

The USART is programmed in standard Smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card 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 will be correctly received this time, by the reprogrammed USART

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

#### Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) 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.

### 44.5.18 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to [Section 44.4: USART implementation on page 1517](#).

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 470](#)).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 Kbps for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the 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 471](#)).
- 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 2 periods will be accepted as a pulse. The IrDA encoder/decoder doesn't work when PSC=0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART\_CR2 register must be configured to '1 stop bit'.

#### IrDA low-power mode

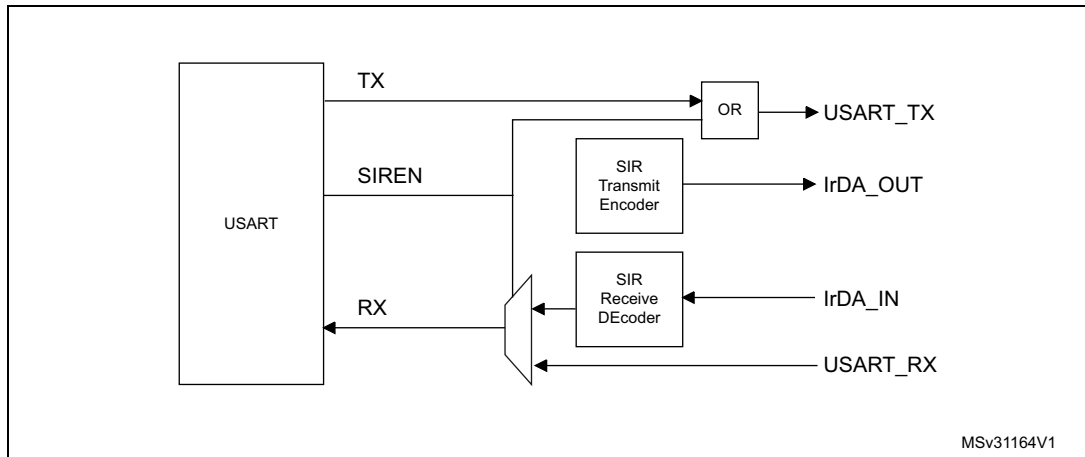
- Transmitter  
In low-power mode, the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally, this value is 1.8432 MHz ( $1.42 \text{ MHz} < \text{PSC} < 2.12 \text{ MHz}$ ). A low-power mode programmable divisor divides the system clock to achieve this value.
- Receiver  
Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART should discard pulses of duration shorter than 1/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).*

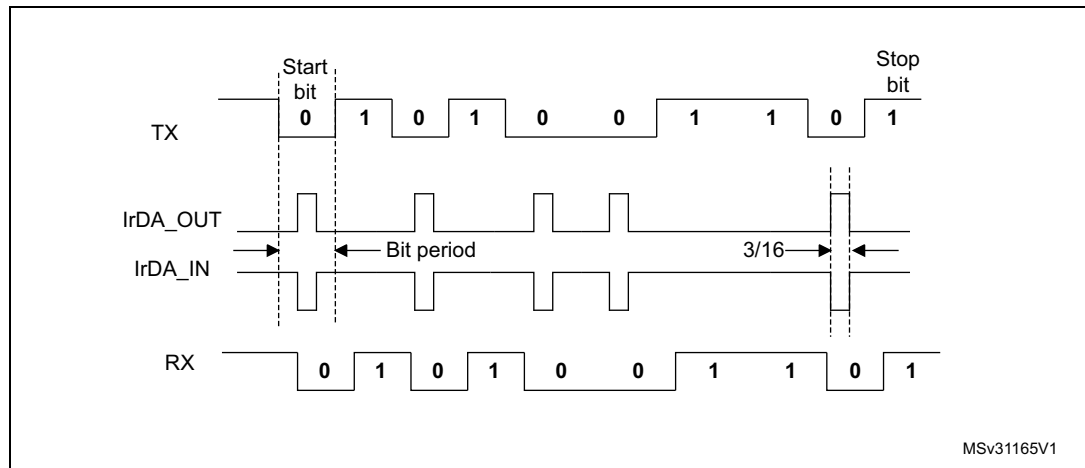


Figure 470. IrDA SIR ENDEC block diagram



MSv31164V1

Figure 471. IrDA data modulation (3/16) - Normal mode



MSv31165V1

### 44.5.19 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 44.4: USART implementation on page 1517](#) to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in [Section 44.5.6](#). To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART\_ISR register.

#### Transmission using DMA

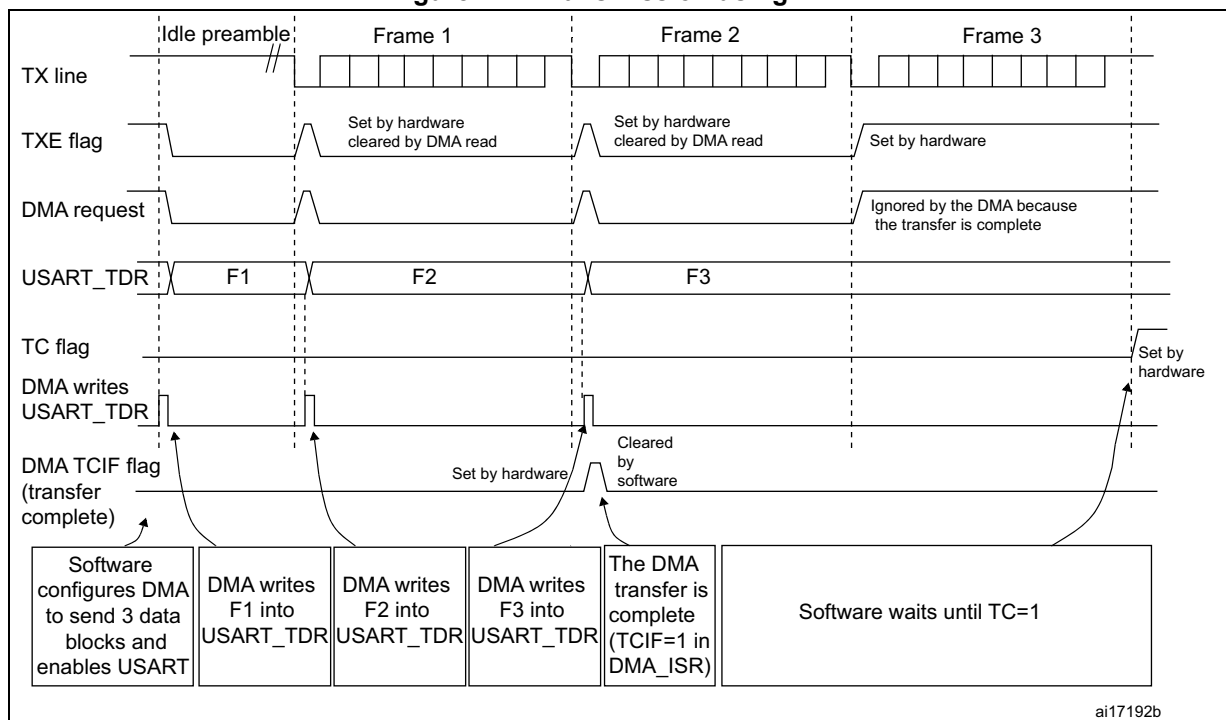
DMA mode can be enabled for transmission by setting DMAT bit in the USART\_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to [Section 8: Direct memory access controller \(DMA\)](#) and [Section 11: Direct memory access controller \(DMA\)](#)) to the USART\_TDR 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 472. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

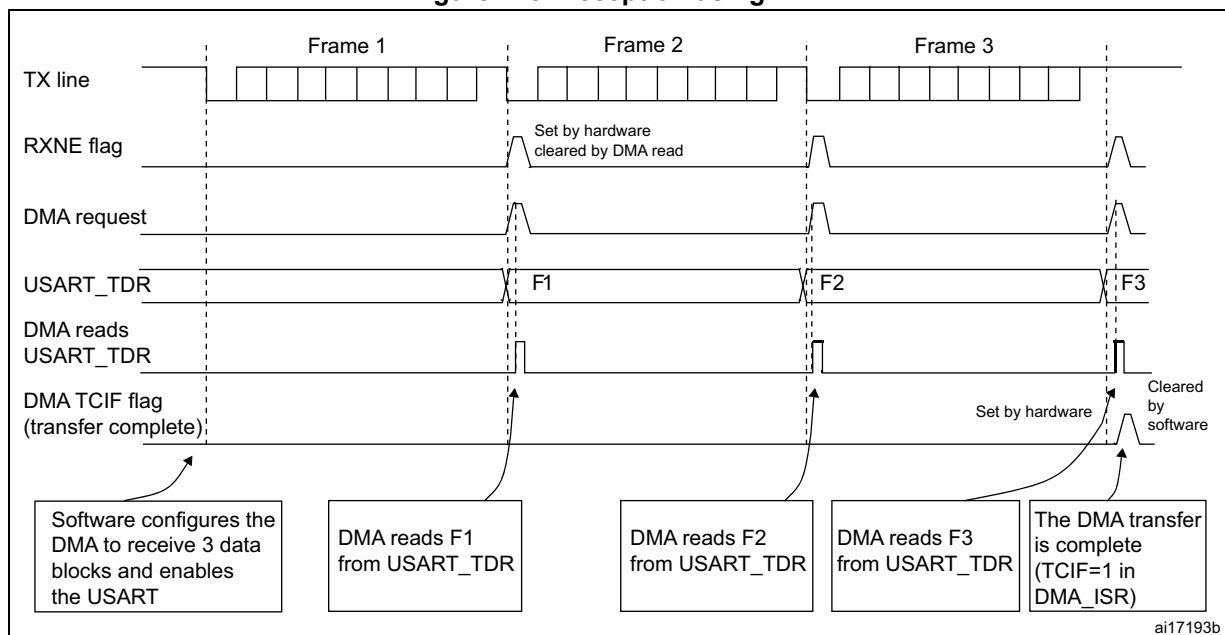
### Reception using DMA

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 [Section 8: Direct memory access controller \(DMA\)](#) and [Section 11: Direct memory access controller \(DMA\)](#)) 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 473. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

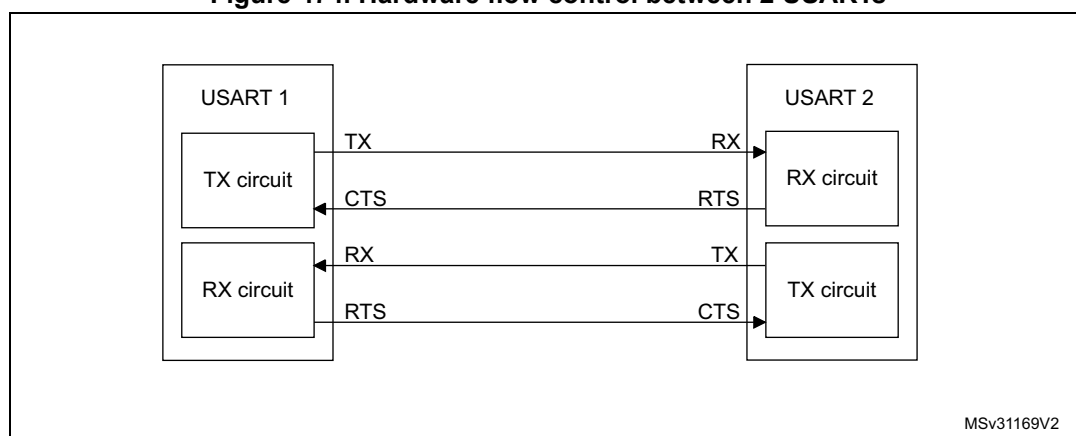
### Error flagging and interrupt generation in multibuffer communication

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.

### 44.5.20 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. The Figure 474 shows how to connect 2 devices in this mode:

Figure 474. Hardware flow control between 2 USARTs

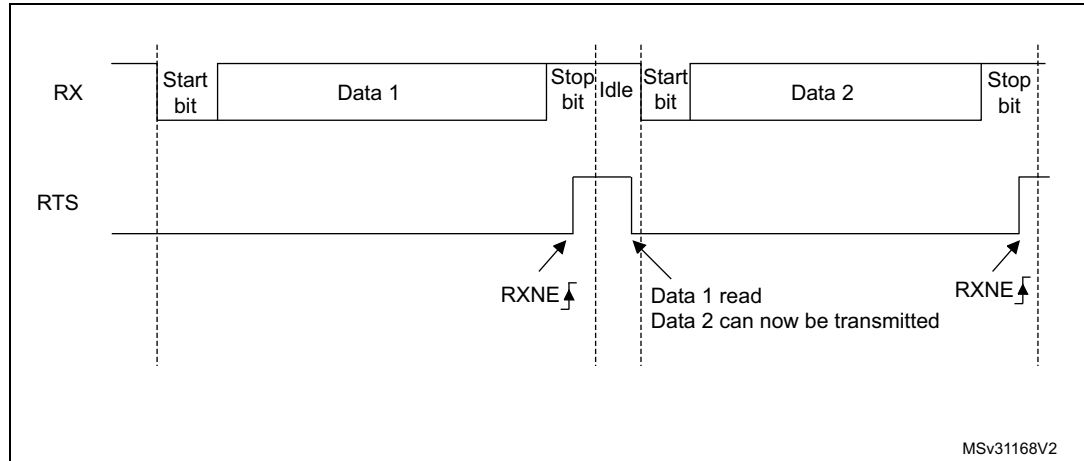


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to '1' in the USART\_CR3 register.

**RS232 RTS flow control**

If the RTS flow control is enabled (RTSE=1), then nRTS is asserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. [Figure 475](#) shows an example of communication with RTS flow control enabled.

**Figure 475. RS232 RTS flow control**



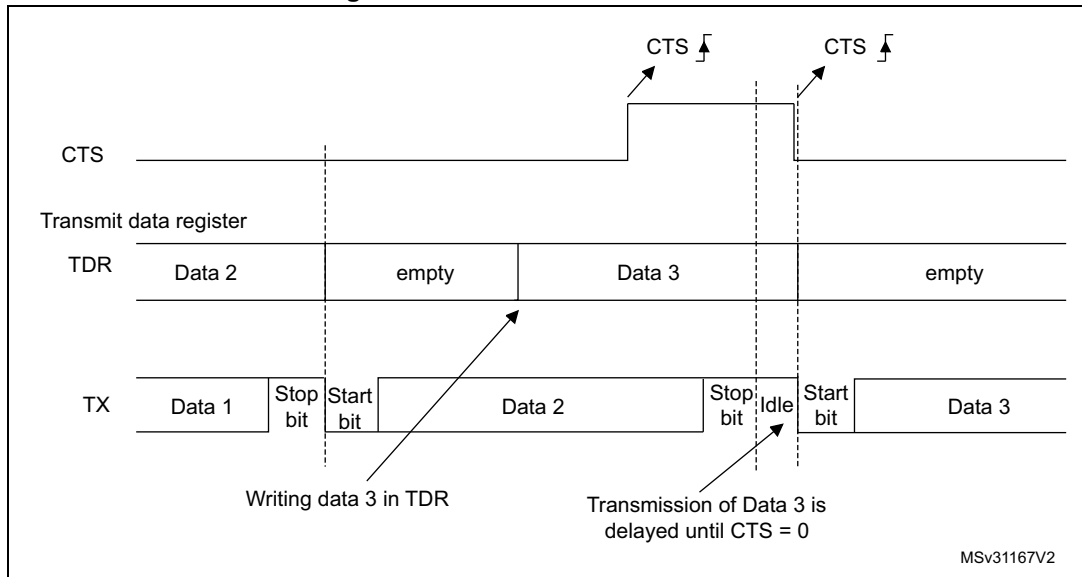
*Note:* When FIFO mode is enabled, nRTS is de-asserted only when RXFIFO is full.

**RS232 CTS flow control**

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When nCTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE=1, the CTSIF status bit is automatically set by hardware as soon as the nCTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART\_CR3 register is set. [Figure 476](#) shows an example of communication with CTS flow control enabled.

Figure 476. RS232 CTS flow control



*Note:* For correct behavior, nCTS must be asserted at least 3 USART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

**RS485 driver enable**

The driver enable feature is enabled by setting bit DEM in the USART\_CR3 control register. This allows the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the USART\_CR1 control register. The de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] 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).

### 44.5.21 USART low-power management

The USART has advanced low-power mode functions, that allow 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 allows 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 allows 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. In this case the `usart_wkup` interrupt is not mandatory and setting the `WUF` being is sufficient to wake up the MCU from low-power mode.

*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 will remain 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 477 shows the USART behavior when the wakeup event is verified.

**Figure 477. Wakeup event verified (wakeup event = address match, FIFO disabled)**

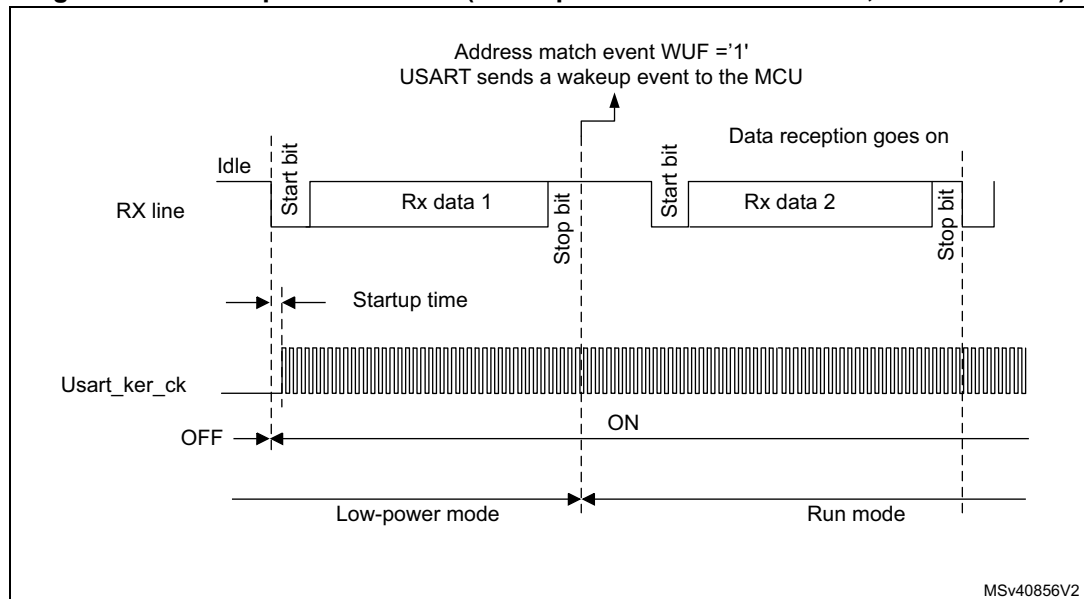
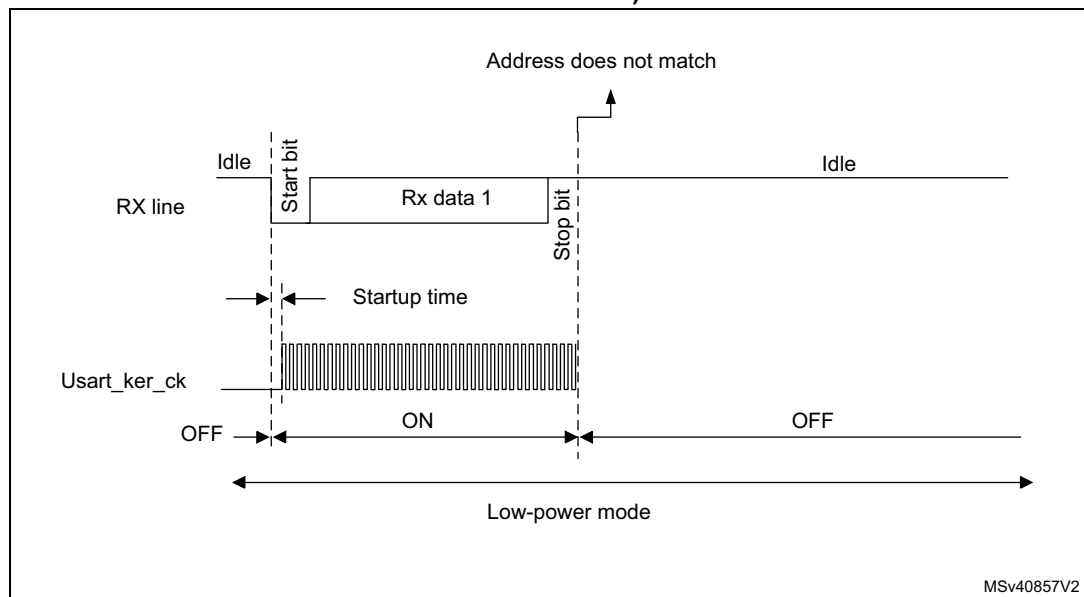


Figure 478 shows the USART behavior when the wakeup event is not verified.

**Figure 478. 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 allows to correctly wake up the microcontroller from low-power mode**

The maximum baud rate allowing to correctly wake up the microcontroller from low-power mode depends on the wakeup time parameter (refer to the device datasheet) and on the USART receiver tolerance (see [Section 44.5.8: 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 266: Tolerance of the USART receiver when BRR \[3:0\] = 0000](#), the USART receiver tolerance equals 3.41%.

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{USART receiver tolerance}$$

$$D_{WUmax} = t_{WUUSART} / (11 \times T_{bit \text{ Min}})$$

$$T_{bit \text{ Min}} = t_{WUUSART} / (11 \times D_{WUmax})$$

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:

$$t_{WUUSART} = 3 \mu\text{s} \text{ (values provided only as examples; for correct values, refer to the device datasheet).}$$

$$D_{WUmax} = 3.41\% - 1\% = 2.41\%$$

$$T_{bit \text{ min}} = 3 \mu\text{s} / (11 \times 2.41\%) = 11.32 \mu\text{s}.$$

As a result, the maximum baud rate allowing to wakeup correctly from low-power mode is:  $1/11.32 \mu\text{s} = 88.36 \text{ Kbaud}$ .

**44.6 USART interrupts**

During USART communications, an interrupt (usart\_it) can be generated by different events. The USART block can also generate a wakeup interrupt (usart\_wkup).

Refer to [Table 269](#) for a detailed description of all USART interrupt requests.

**Table 269. USART interrupt requests**

Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Interrupt activated	
				usart_it	usart_wkup
Transmit data register empty	TXE	TXEIE	TXE cleared when a data is written in TDR	YES	NO
Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFNF cleared when TXFIFO is full.	YES	NO
Transmit FIFO Empty	TXFE	TXFEIE	TXFE cleared when the TXFIFO contains at least one data or by setting TXFRQ bit.	YES	YES



Table 269. USART interrupt requests (continued)

Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Interrupt activated	
				usart_it	usart_wkup
Transmit FIFO threshold reached	TXFT	TXFTIE	TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold	YES	YES
CTS interrupt	CTSIF	CTSIE	CTSIF cleared by software by setting CTSCF bit.	YES	NO
Transmission Complete	TC	TCIE	TC cleared when a data is written in TDR or by setting TCCF bit.	YES	NO
Transmission Complete Before Guard Time	TCBGT	TCBGTIE	TCBGT cleared when a data is written in TDR or by setting TCBGTCF bit.	YES	NO
Receive data register not empty (data ready to be read)	RXNE	RXNEIE	RXNE cleared by reading RDR or by setting RXFRQ bit.	YES	YES
Receive FIFO Not Empty	RXFNE	RXFNEIE	RXFNE cleared when the RXFIFO is empty or by setting RXFRQ bit.	YES	YES
Receive FIFO Full	RXFF <sup>(1)</sup>	RXFFIE	RXFF cleared when the RXFIFO contains at least one data.	YES	YES
Receive FIFO threshold reached	RXFT	RXFTIE	RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold	YES	YES
Overrun error detected	ORE	RX-NEIE/RX-FNEIE	ORE cleared by setting ORECF bit.	YES	NO
Idle line detected	IDLE	IDLEIE	IDLE cleared by setting IDLECF bit.	YES	NO
Parity error	PE	PEIE	PE cleared by setting PECF bit.	YES	NO
LIN break	LBDF	LBDIE	LBDF cleared by setting LBDCF bit.	YES	NO
Noise Flag, Overrun error and Framing Error in multibuffer communication.	NE or ORE or FE	EIE	NE cleared by setting NCF bit. ORE cleared by setting ORECF bit. FE flag cleared by setting FECF bit.	YES	NO
Character match	CMF	CMIE	CMF cleared by setting CMCF bit.	YES	NO
Receiver timeout	RTOF	RTOFIE	RTOF cleared by setting RTOCCF bit.	YES	NO

**Table 269. USART interrupt requests (continued)**

Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Interrupt activated	
				usart_it	usart_wkup
End of Block	EOBF	EOBIE	EOBF is cleared by setting EOBCF bit.	YES	NO
Wakeup from low-power mode	WUF	WUFIE	WUF is cleared by setting WUCF bit.	YES	YES
SPI slave underrun error	UDR	EIE	UDR is cleared by setting UDRCF bit.	YES	NO
Transmit FIFO threshold reached	TXFT	TXFTIE	TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold	YES	YES
Receive FIFO threshold reached	RXFT	RXFTIE	RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold.	YES	YES

1. 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 will not be written and once n data are received and written in the RXFIFO, the RXFF interrupt will be asserted (RXFF flag is not set).

## 44.7 USART registers

Refer to [Section 1.2 on page 75](#) for a list of abbreviations used in register descriptions.

### 44.7.1 USART control register 1 (USART\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXF FIE	TXFEIE	FIFO EN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE/ TXFNFI E	TCIE	RXNEIE /RXFNE IE	IDLEIE	TE	RE	UESM	UE
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Bit 31 **RXFFIE**: RXFIFO Full interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when RXFF=1 in the USART\_ISR register

*Note: When FIFO mode is disabled, this bit is reserved and must be kept at reset value.*

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

*Note: When FIFO mode is disabled, this bit is reserved and 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 EOBIF flag is set in the USART\_ISR register

*Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to [Section 44.4: USART implementation on page 1517](#).*

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 44.4: USART implementation on page 1517](#).*

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 44.4: USART implementation on page 1517](#).*

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 44.4: USART implementation on page 1517](#).*

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 will be 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/TXFNFIE**: Transmit data register empty/TXFIFO not full interrupt enable  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever TXE/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 **RXNEIE/RXFNEIE**: Receive data register not empty/RXFIFO not empty interrupt enable  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever ORE=1 or RXNE/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 44.4: USART implementation on page 1517](#).*

**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 SCLK is always available when CLKEN = 1, regardless of the UE bit value.*



### 44.7.2 USART control register 2 (USART\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w		r/w	r/w	r/w	r/w			r/w

Bits 31:28 **ADD[7:4]**: Address of the USART node

This bitfield gives the address of the USART node or a character code to be recognized. It is used to wake up the MCU with 7-bit address mark detection in multiprocessor communication during Mute mode or low-power mode. The MSB of the character sent by the transmitter should be equal to 1. It can also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on match. This bitfield can only be written when reception is disabled (RE = 0) or the USART is disabled (UE=0)

Bits 27:24 **ADD[3:0]**: Address of the USART node

This bitfield gives the address of the USART node or a character code to be recognized. This is used for wakeup with address mark detection, in multiprocessor communication during Mute mode or low-power mode. This bitfield can only be written when reception is disabled (RE = 0) or the USART is disabled (UE=0)

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.  
 0: Receiver timeout feature disabled.  
 1: Receiver timeout feature enabled.  
 When this feature is enabled, the RTOF flag in the 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 44.4: USART implementation on page 1517.*

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 -> 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 44.4: USART implementation on page 1517.*

**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 44.4: USART implementation on page 1517](#).*

**Bit 19 MSBFIRST:** Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the USART is disabled (UE=0).

**Bit 18 DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)

1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.

This bitfield can only be written when the USART is disabled (UE=0).

**Bit 17 TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels ( $V_{DD}$  =1/idle, Gnd=0/mark)

1: TX pin signal values are inverted. ( $V_{DD}$  =0/mark, Gnd=1/idle).

This allows the use of an external inverter on the TX line.

This bitfield can only be written when the USART is disabled (UE=0).

**Bit 16 RXINV:** RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels ( $V_{DD}$  =1/idle, Gnd=0/mark)

1: RX pin signal values are inverted. ( $V_{DD}$  =0/mark, Gnd=1/idle).

This allows 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 allows to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the USART is disabled (UE=0).

**Bit 14 LINEN:** LIN mode enable

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART\_CR1 register, and to detect LIN Sync breaks.

This bitfield can only be written when the USART is disabled (UE=0).

*Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value. Refer to [Section 44.4: USART implementation on page 1517](#).*

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 allows the user to enable the SCLK pin.

0: SCLK pin disabled

1: SCLK pin enabled

This bit can only be written when the USART is disabled (UE=0).

*Note: If neither synchronous mode nor Smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to [Section 44.4: USART implementation on page 1517](#).*

*In Smartcard mode, in order to provide correctly the SCLK 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 allows the user to select the polarity of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on SCLK pin outside transmission window

1: Steady high value on SCLK pin outside transmission window

This bit can only be written when the USART is disabled (UE=0).

*Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 44.4: USART implementation on page 1517](#).*

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see [Figure 458](#) and [Figure 459](#))

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 44.4: USART implementation on page 1517](#).*

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the SCLK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the SCLK pin

1: The clock pulse of the last data bit is output to the SCLK pin

**Caution:** The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M 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 44.4: USART implementation on page 1517](#).*

## 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 44.4: USART implementation on page 1517.*

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 44.4: USART implementation on page 1517.*

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**:  
 When the DSI\_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 44.4: USART implementation on page 1517.*

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 44.4: USART implementation on page 1517.*

*Note: The CPOL, CPHA and LBCL bits should not be written while the transmitter is enabled.*

### 44.7.3 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.
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w



Bits 31:29 **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 44.4: USART implementation on page 1517](#).*

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 44.4: USART implementation on page 1517](#).*

- 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 44.4: USART implementation on page 1517](#).*
- 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 44.4: USART implementation on page 1517](#).*
- 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 44.4: USART implementation on page 1517](#).*
- Bit 14 **DEM**: Driver enable mode
- This bit allows the user to activate the external transceiver control, through the DE signal.
- 0: DE function is disabled.
- 1: DE function is enabled. The DE signal is output on the RTS pin.
- This bit can only be written when the USART is disabled (UE=0).
- Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. [Section 44.4: USART implementation on page 1517](#).*
- Bit 13 **DDRE**: DMA Disable on Reception Error
- 0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data will be transferred. (used for Smartcard mode)
- 1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE (RXFNE is 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 will be bypassed and data will be 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 allows checking the communication flow w/o reading the data

**Bit 11 ONEBIT:** One sample bit method enable

This bit allows the user to select the sample method. When the one sample bit method is selected the noise detection flag (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 44.4: USART implementation on page 1517](#).

**Bit 9 CTSE:** CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is deasserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.

This bit can only be written when the USART is disabled (UE=0)

*Note:* If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 44.4: USART implementation on page 1517](#).

**Bit 8 RTSE:** RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE=0).

*Note:* If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 44.4: USART implementation on page 1517](#).

**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 44.4: USART implementation on page 1517](#).*

**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 44.4: USART implementation on page 1517](#).*

**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 44.4: USART implementation on page 1517](#).*

**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 44.4: USART implementation on page 1517](#).*

**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.



### 44.7.4 USART baud rate register (USART\_BRR)

This register can only be written when the USART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:4 **BRR[15:4]**

$$BRR[15:4] = USARTDIV[15:4]$$

Bits 3:0 **BRR[3:0]**

When OVER8 = 0,  $BRR[3:0] = USARTDIV[3:0]$ .

When OVER8 = 1:

$BRR[2:0] = USARTDIV[3:0]$  shifted 1 bit to the right.

$BRR[3]$  must be kept cleared.

### 44.7.5 USART guard time and prescaler register (USART\_GTPR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]								PSC[7:0]							
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **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 44.4: USART implementation on page 1517.*

Bits 7:0 **PSC[7:0]**: Prescaler value

**In IrDA low-power and normal IrDA mode:**

PSC[7:0] = IrDA Normal and Low-Power Baud Rate

Used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

**In Smartcard mode:**

PSC[4:0]: Prescaler value

Used for programming the prescaler for dividing the USART source clock to provide the Smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

This 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 44.4: USART implementation on page 1517.*

### 44.7.6 USART receiver timeout register (USART\_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLEN[7:0]								RTO[23:16]							
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:24 **BLLEN[7:0]**: Block Length

This bitfield gives the Block length in Smartcard T=1 Reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1.

Examples:

BLLEN = 0 -> 0 information characters + LEC

BLLEN = 1 -> 0 information characters + CRC

BLLEN = 255 -> 254 information characters + CRC (total 256 characters)

In Smartcard mode, the Block length counter is reset when TXE=0 (TXFE = 0 in case FIFO mode is enabled).

This bitfield can be used also in other modes. In this case, the Block length counter is reset when RE=0 (receiver disabled) and/or when the EOBCF bit is written to 1.

*Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.*

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bitfield gives the Receiver timeout value in terms of number of baud clocks.

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 44.4: USART implementation on page 1517](#).*

**44.7.7 USART request register (USART\_RQR)**

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	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 allows 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 allows 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 flag 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 44.4: USART implementation on page 1517](#).*

### 44.7.8 USART interrupt and status register (USART\_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 00 if FIFO is disabled

XX = 28 if FIFO/Smartcard mode is enabled

XX = 08 if FIFO is enabled and Smartcard mode is disabled

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	TXE/TX FNF	TC	RXNE/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 will be 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 will 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 44.4: USART implementation on page 1517](#).*

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 44.4: USART implementation on page 1517](#).*
- 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 44.4: USART implementation on page 1517](#).*
- 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 44.4: USART implementation on page 1517](#).*
- 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: No break character is transmitted  
1: Break character will be transmitted
- 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 will also be set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation was completed without success (ABRE=1) (ABRE, RXNE and FE are also set in this case)  
It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the 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\_CR3 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 44.4: USART implementation on page 1517](#).*

**Bit 12 EOBF:** End of block flag

This bit is set by hardware when a complete block has been received (for example T=1 Smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.  
An interrupt is generated if the EOBI=1 in the USART\_CR2 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 44.4: USART implementation on page 1517](#).*

**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 will be 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 nCTS input pin.

0: nCTS line set

1: nCTS line reset

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

**Bit 9 CTSIF:** CTS interrupt flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART\_ICR register.

An interrupt is generated if CTSIE=1 in the USART\_CR3 register.

0: No change occurred on the nCTS status line

1: A change occurred on the nCTS status line

*Note: 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 44.4: USART implementation on page 1517](#).*

**Bit 7 TXE/TXFNF:** Transmit data register empty/TXFIFO not full

When the FIFO mode is disabled, 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).

When the FIFO mode is enabled, 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 TXEIE/TXFNFIE bit =1 in the USART\_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/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 will be 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 TXE/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 will be set immediately.*



**Bit 5 RXNE/RXFNE:** Read data register not empty/RXFIFO 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.

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.

RXNE/RXFNE is cleared when the RXFIFO is empty. The RXNE/RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART\_RQR register.

An interrupt is generated if RXNEIE/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 will not be set again until the RXNE bit has been set (i.e. a new idle line occurs).*

*If Mute mode is enabled (MME=1), IDLE is set if the USART is not mute (RWU=0), whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.*

**Bit 3 ORE:** Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART\_RDR register while RXNE=1 (RXFF = 1 in case FIFO mode is enabled). It is cleared by a software, writing 1 to the ORECF, in the USART\_ICR register.

An interrupt is generated if RXNEIE/ RXFNEIE=1 or EIE = 1 in the USART\_CR1 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 NFCF 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/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 44.5.8: Tolerance of the USART receiver to clock deviation on page 1535).*

*In FIFO mode, 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\_CR1 register.

- 0: No Framing error is detected
- 1: Framing error or break character is detected

*Note: In FIFO mode, 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 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

*Note: In FIFO mode, this error is associated with the character in the USART\_RDR.*

### 44.7.9 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	IDLECF	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 44.4: USART implementation on page 1517.*

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 44.4: USART implementation on page 1517](#)*

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 44.4: USART implementation on page 1517](#).*

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 44.4: USART implementation on page 1517](#).*

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 44.4: USART implementation on page 1517](#).*

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 44.4: USART implementation on page 1517](#).*

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.

### 44.7.10 USART receive data register (USART\_RDR)

Address offset: 0x24

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value

Contains the received data character.

The RDR register provides the parallel interface between the input shift register and the internal bus (see [Figure 452](#)).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

### 44.7.11 USART transmit data register (USART\_TDR)

Address offset: 0x28

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The USART\_TDR register provides the parallel interface between the internal bus and the output shift register (see [Figure 452](#)).

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.*

### 44.7.12 USART prescaler register (USART\_PRESC)

This register can only be written when the USART is disabled (UE=0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												r/w	r/w	r/w	r/w

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 will be '1011' i.e. input clock divided by 256.*

### 44.7.13 USART register map

The table below gives the USART register map and reset values.

**Table 270. USART 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	USART_CR1	RXFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT4	DEAT3	DEAT2	DEAT1	DEAT0	DEDT4	DEDT3	DEDT2	DEDT1	DEDT0	OVER8	CMIE	MME	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x04	USART_CR2	ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD1	ABRMOD0	ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP [1:0]	CLKEN	CPOL	CPHA	LBCL	Res.	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
0x08	USART_CR3	TXFT CFG			RXFTE	RXFT CFG			TCBGTIE	TXFTIE	WUFIE	WUS [1:0]	SCAR CNT2:0]		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
0x0C	USART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_GTPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]					PSC[7:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USART_RTOR	BLEN[7:0]							RTO[23:0]																									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																	
0x1C	USART_ISR	Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE	
	Reset value					0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF	
	Reset value												0			0				0	0	0	Res.	0	0	0	0	0	0	0	0	0	0	0
0x24	USART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]									
	Reset value																								0	0	0	0	0	0	0	0	0	0
0x28	USART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]									
	Reset value																								0	0	0	0	0	0	0	0	0	0

Table 270. 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				
0x2C	USART_PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value																																		PRESCALE R[3:0]		
																																		0	0	0	0

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.

## 45 Low-power universal asynchronous receiver transmitter (LPUART)

This section describes the low-power universal asynchronous receiver transmitter (LPUART).

### 45.1 LPUART introduction

The LPUART is an UART which allows bidirectional UART communications with a limited power consumption. Only 32.768 kHz LSE clock is required to allow UART communications up to 9600 baud/s. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller 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.



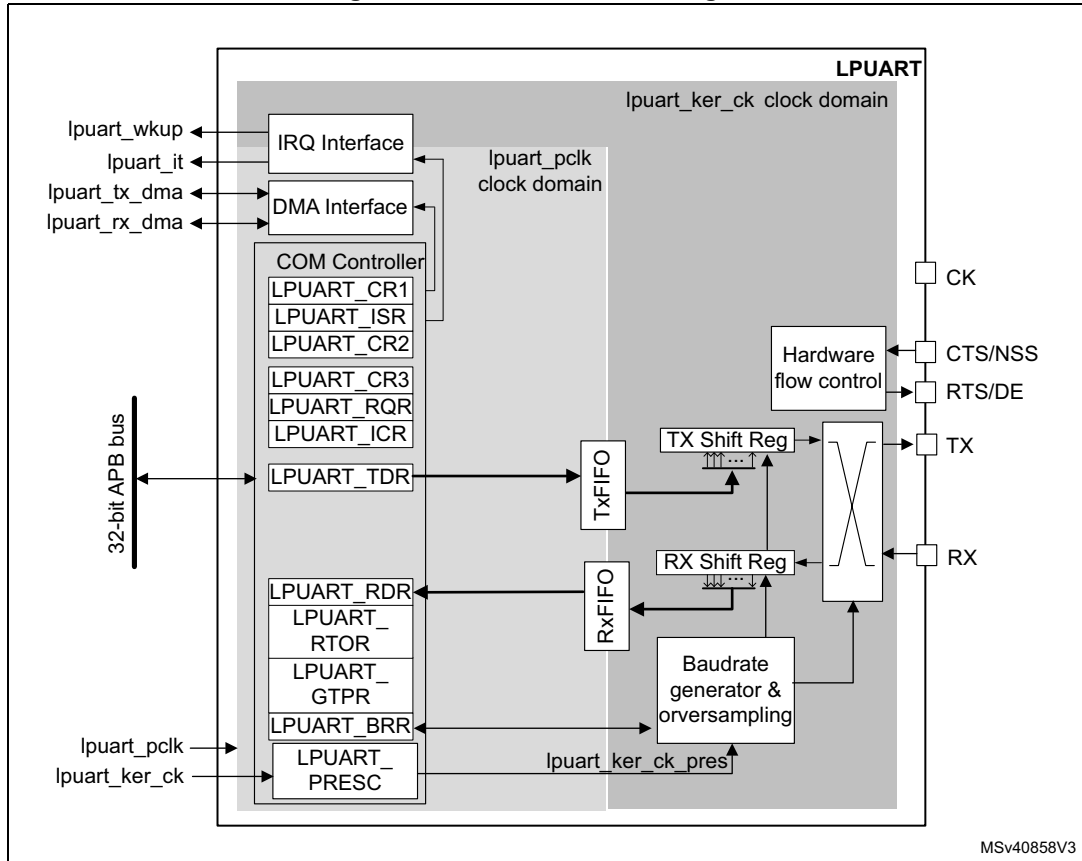
## 45.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 baud/s to 9600 baud/s 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

### 45.3 LPUART functional description

#### 45.3.1 LPUART block diagram

Figure 479. LPUART block diagram



The simplified block diagram given in [Figure 479](#) 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.

### 45.3.2 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.

### 45.3.3 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 453](#)).

- 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.

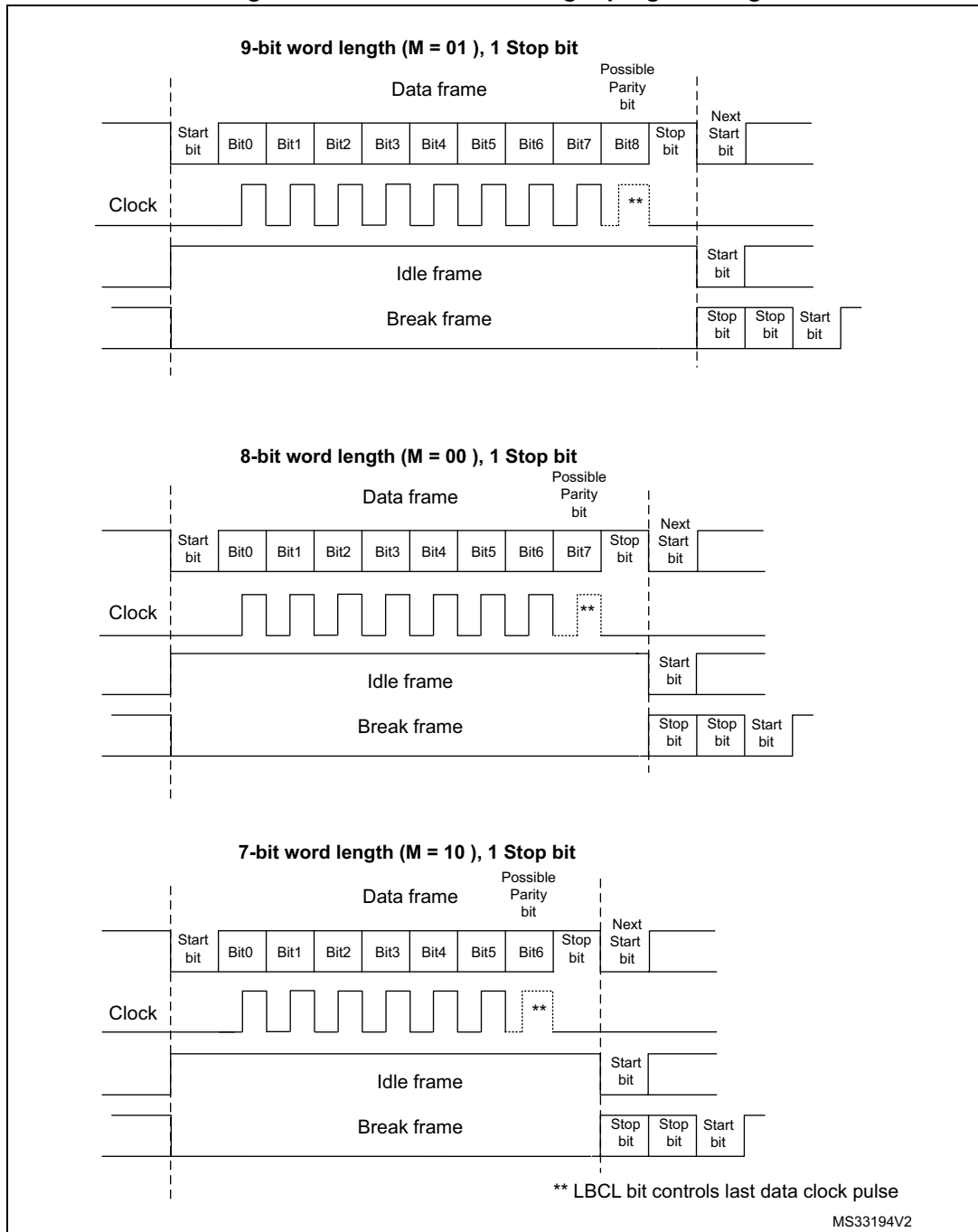
An **Idle character** is interpreted as an entire frame of "1"s. (The number of "1" 's will include the number of stop bits).

A **Break character** is interpreted on receiving "0"s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The 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 480. LPUART word length programming



### 45.3.4 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 9 bits the maximum data word length is 9 bits, the TXFIFO is 9-bits 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 Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bitfields.

In this case, the RXFT flag is set in the LPUART\_ISR register. This means that RXFTCFG data have been received: 1 data in LPUART\_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to '101', the RXFT flag will be 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 will not set the overrun flag.

- The Tx interrupt is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bitfields.

### 45.3.5 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 479](#)).

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 will corrupt the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost.

An idle frame will be sent after the TE bit is enabled.

#### Configurable stop bits

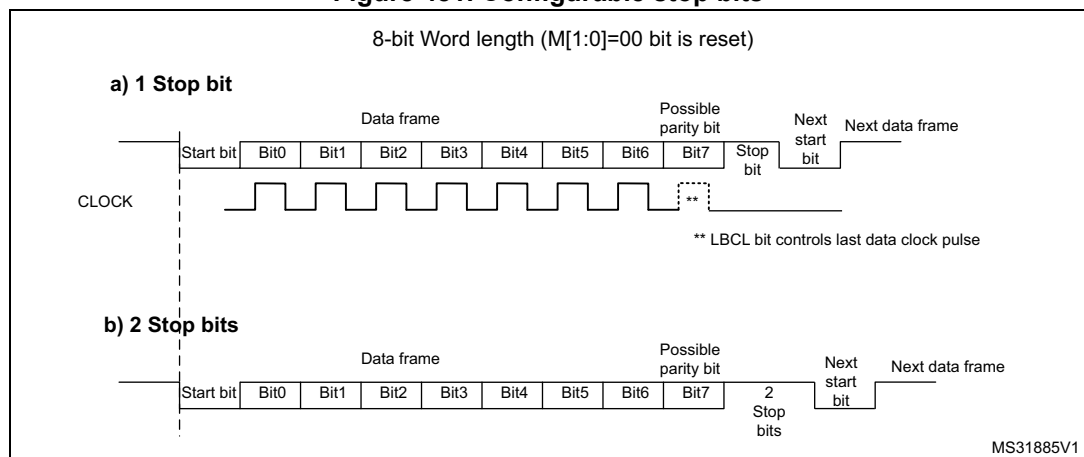
The number of stop bits to be transmitted with every character can be programmed in LPUART\_CR2 (bits 13,12).

- **1 stop bit:** This is the default value of number of stop bits.
- **2 Stop bits:** This will be supported by normal LPUART, Single-wire and Modem modes.

An idle frame transmission will include the stop bits.

A break transmission will be 10 low bits (when M[1:0] = '00') or 11 low bits (when M[1:0] = '01') or 9 low bits (when M[1:0] = '10') followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

**Figure 481. 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 44.5.10: 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.

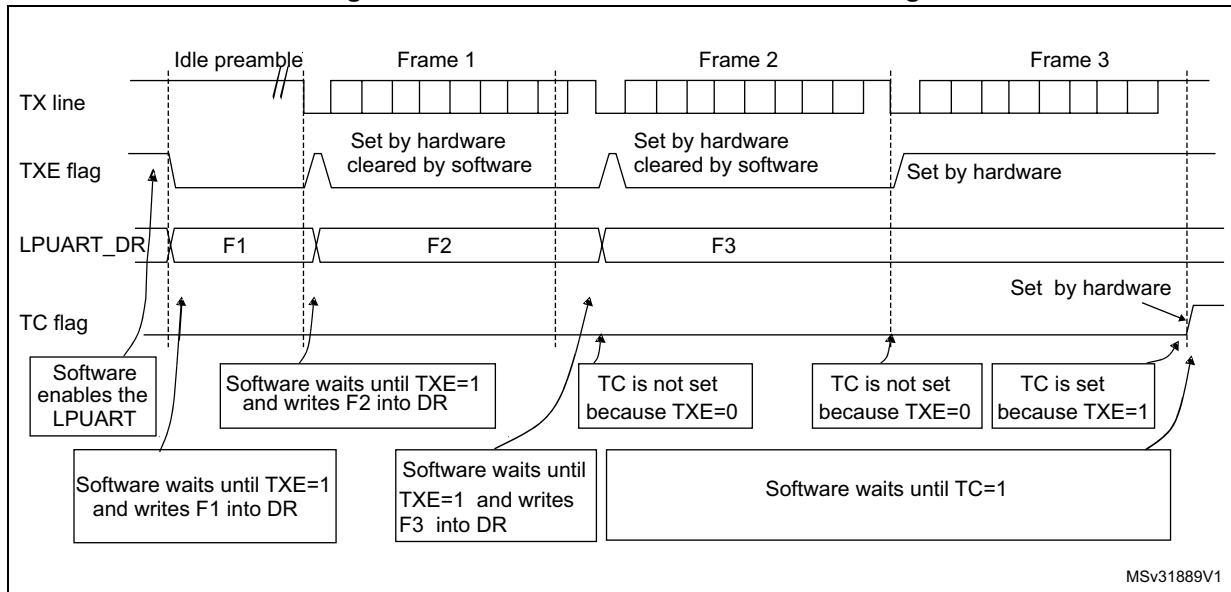
When the TXFIFO is not full, the TXFNF flag stays at '1' even after a write in LPUART\_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if TXFNEIE bit is set.

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 microcontroller to enter the low-power mode (see [Figure 482: TC/TXE behavior when transmitting](#)).

Figure 482. 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 480).

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.

**45.3.6 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 44.5.10: 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 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 when a character is received when RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.

An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  - the ORE bit is set;
  - the RDR content will not be lost. The previous data is available when a read to LPUART\_RDR is performed.;
  - the shift register will be 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 will not be lost. It is available when a read to LPUART\_RDR is performed.
  - the shift register will be overwritten. After that, any data received during overrun is lost.
  - an interrupt is generated if either the RXFNEIE bit or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.

*Note:* The ORE bit, when set, indicates that at least 1 data has been lost. T

When the FIFO mode is disabled, there are two possibilities

- if RXNE=1, then the last valid data is stored in the receive register (RDR) and can be read,
- if RXNE=0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.

**Selecting the clock source**

The choice of the clock source is done through the Clock Control system (see Section Reset and clock controller (RCC)). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:

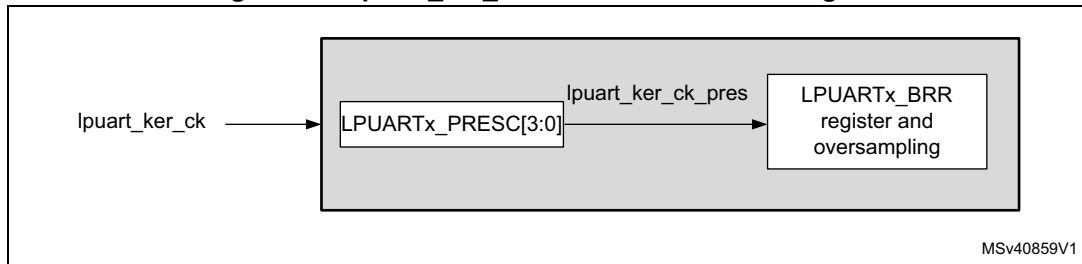
- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is lpuart\_ker\_ck.

When the dual clock domain and the wakeup from low-power mode features are supported, the lpuart\_ker\_ck clock source can be configured in the RCC (see Section Reset and clock controller (RCC)). Otherwise, the lpuart\_ker\_ck is the same as lpuart\_pclk.

The lpuart\_ker\_ck can be divided by a programmable factor in the LPUART\_PRESC register.

**Figure 483. lpuart\_ker\_ck clock divider block diagram**



Some lpuart\_ker\_ck sources allow 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 allow LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming baud as close as possible to the middle of the baud-period. Only a single sample is taken of each of the incoming bauds.

*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 will be 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.

## 45.3.7 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART\_BRR register.

$$\text{Tx/Rx baud} = \frac{256 \times \text{Lpuartckpres}}{\text{LPUARTDIV}}$$

LPUARTDIV is defined in the LPUART\_BRR register.

*Note:* The baud counters are updated to the new value in the baud registers after a write operation to LPUART\_BRR. Hence the baud rate register value should not be changed during communication.

*It is forbidden to write values lower than 0x300 in the LPUART\_BRR register.*

*$f_{CK}$  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 271. Error calculation for programmed baud rates at lpuart\_ker\_ck\_pres= 32,768 KHz

Baud rate		lpuart_ker_ck_pres= 32,768 KHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate
1	0.3 Kbps	0.3 Kbps	0x6D3A	0
2	0.6 Kbps	0.6 Kbps	0x369D	0
3	1200 Bps	1200.087 Bps	0x1B4E	0.007
4	2400 Bps	2400.17 Bps	0xDA7	0.007
5	4800 Bps	4801.72 Bps	0x6D3	0.035
6	9600 Kbps	9608.94 Bps	0x369	0.093

Table 272. Error calculation for programmed baud rates at f<sub>CK</sub> = 100 MHz

Baud rate		f <sub>CK</sub> = 100MHz		
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	30000 Kbaud	33032258,06 Baud	307	0,1

### 45.3.8 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 273](#):

- Number of Stop bits defined through STOP[1:0] bits in the LPUART\_CR2 register
- LPUART\_BRR register value.

**Table 273. 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 273](#) 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').

### 45.3.9 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, with their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant LPUART service overhead for all non addressed receivers.

The non addressed devices can be placed in Mute mode by means of the muting function. To use the Mute mode feature, the MME bit must be set in the LPUART\_CR1 register.

*Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two lpuart\_ker\_ck cycles), otherwise Mute mode might remain active.*

When the Mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in LPUART\_ISR register is set to '1'. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART\_RQR register, under certain conditions.

The LPUART can enter or exit from Mute mode using one of two methods, depending on the WAKE bit in the LPUART\_CR1 register:

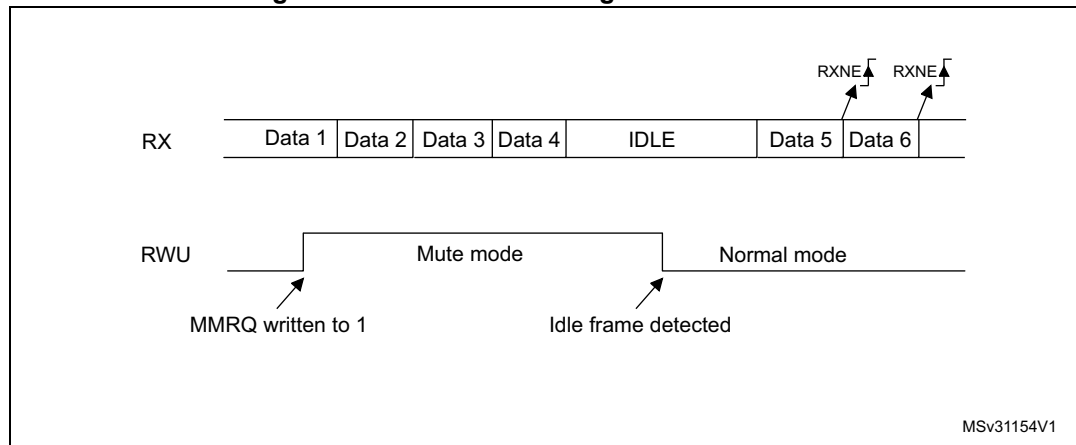
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

**Idle line detection (WAKE=0)**

The LPUART enters Mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The LPUART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the LPUART\_ISR register. An example of Mute mode behavior using Idle line detection is given in [Figure 484](#).

**Figure 484. Mute mode using Idle line detection**



*Note: If the MMRQ is set while the IDLE character has already elapsed, Mute mode will not be entered (RWU is not set). If the 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.

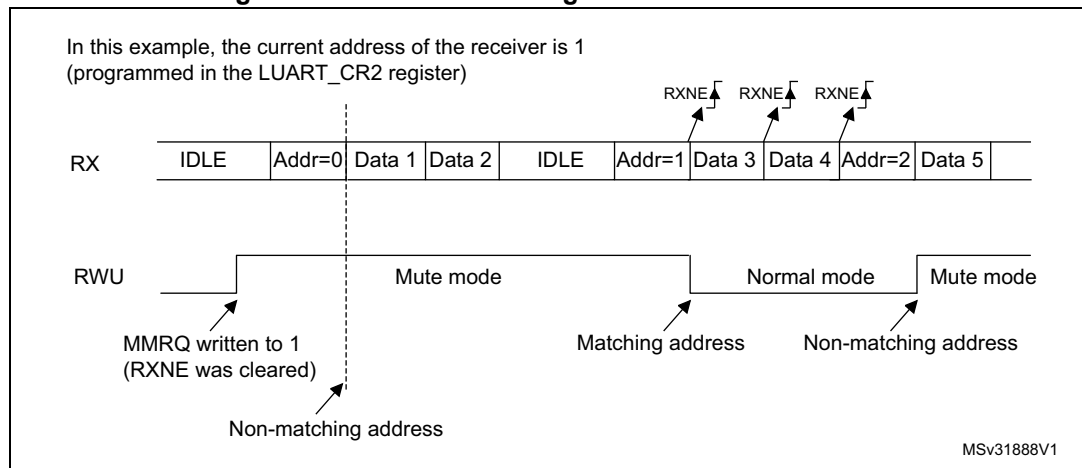
The LPUART also enters Mute mode when the MMRQ bit is written to '1'. The RWU bit is also automatically set in this case.

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 485](#).

**Figure 485. Mute mode using address mark detection**



### 45.3.10 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 274](#).

**Table 274: 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



Table 274: LPUART frame formats

M bits	PCE bit	LPUART frame <sup>(1)</sup>
10	0	SB   7bit data   STB
10	1	SB   6-bit data   PB   STB

- Legends: SB: start bit, STB: stop bit, PB: parity bit.
- In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

### Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit will be 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=00110101 and 4 bits set, then the parity bit will be 1 if odd parity is selected (PS bit in LPUART\_CR1 = 1).

### Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART\_ISR register and an interrupt is generated if PEIE is set in the LPUART\_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART\_ICR register.

### Parity generation in transmission

If the PCE bit is set in LPUART\_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

## 45.3.11 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.

### 45.3.12 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 44.4: USART implementation on page 1517](#) to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in [Section 44.5.6](#). To perform continuous communication. When FIFO is disabled, you can clear the TXE/ RXNE flags in the LPUART\_ISR register.

#### Transmission using DMA

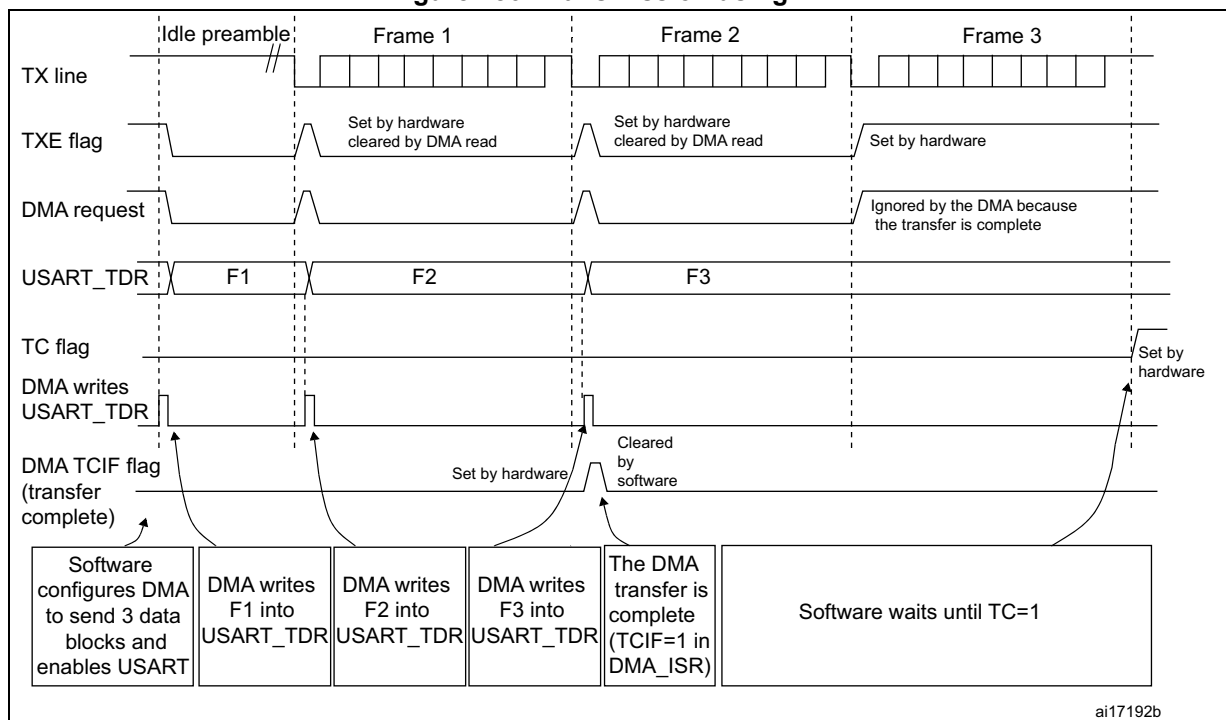
DMA mode can be enabled for transmission by setting DMAT bit in the LPUART\_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the [Direct memory access controller sections](#)) to the LPUART\_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) 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 486. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

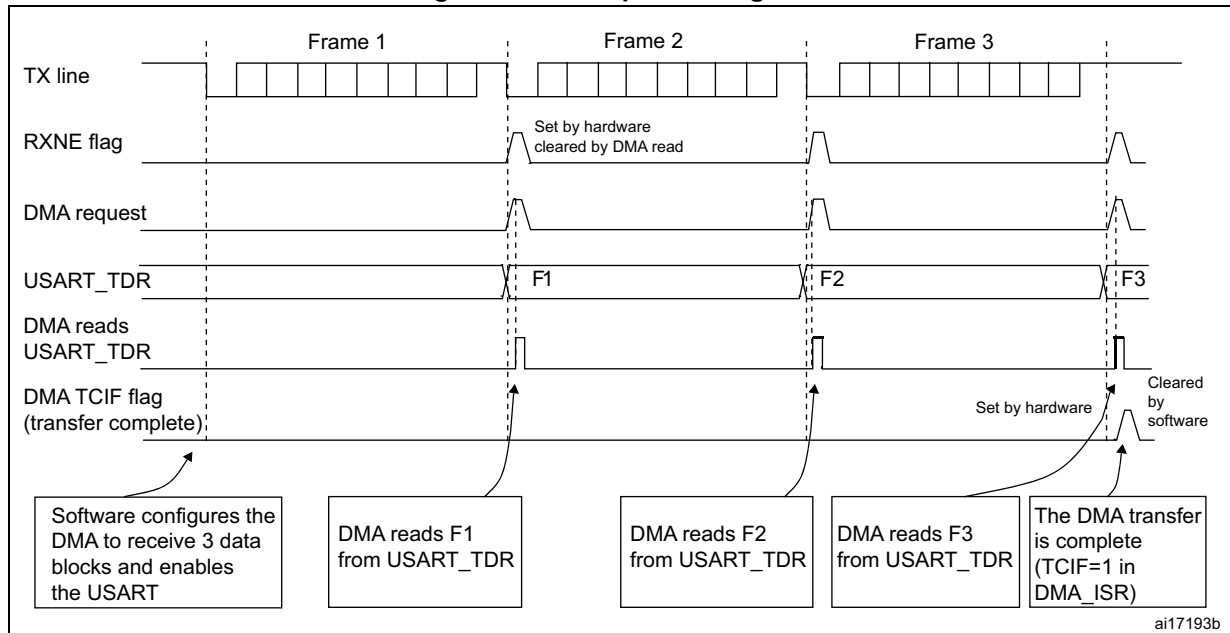
### Reception using DMA

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 [Direct memory access controller \(DMA\) sections](#)) 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 487. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

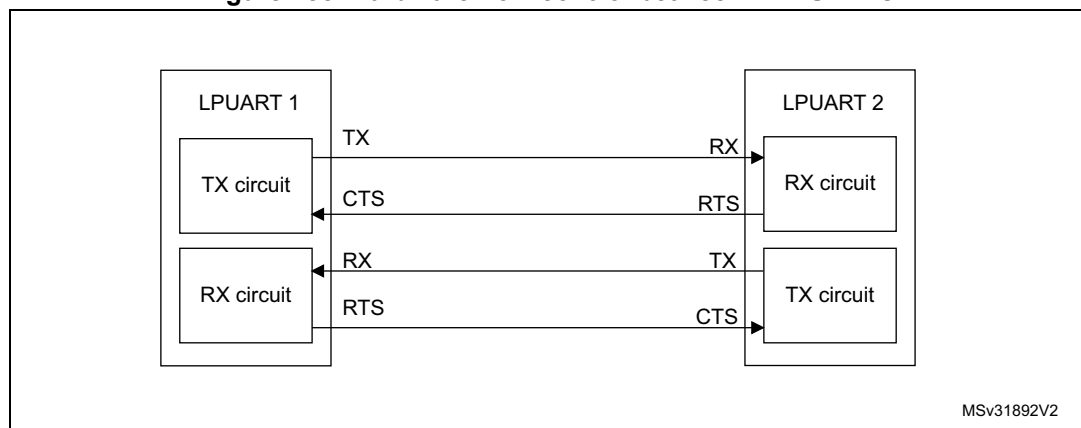
### Error flagging and interrupt generation in multibuffer communication

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.

### 45.3.13 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. The Figure 474 shows how to connect 2 devices in this mode:

Figure 488. Hardware flow control between 2 LPUARTs

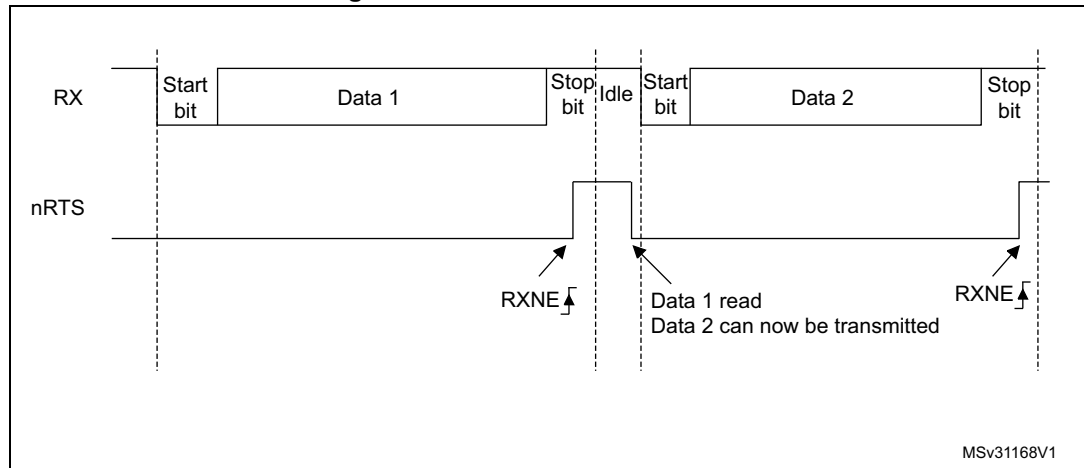


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART\_CR3 register).

**RS232 RTS flow control**

If the RTS flow control is enabled (RTSE=1), then nRTS is asserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. [Figure 489](#) shows an example of communication with RTS flow control enabled.

**Figure 489. RS232 RTS flow control**



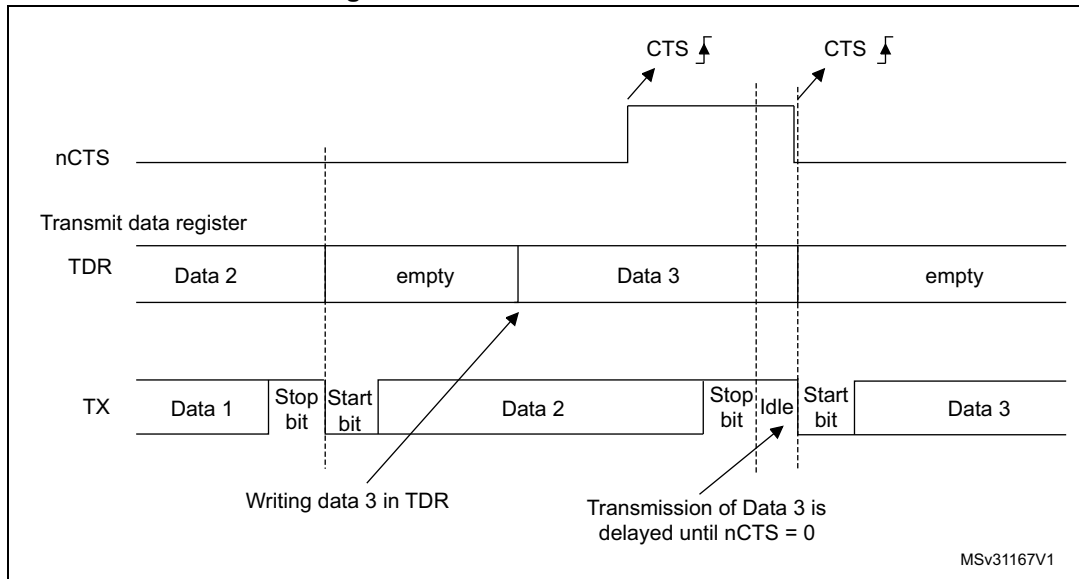
*Note:* When FIFO mode is enabled, nRTS is de-asserted only when RXFIFO is full.

**RS232 CTS flow control**

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When nCTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE=1, the CTSIF status bit is automatically set by hardware as soon as the nCTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART\_CR3 register is set. [Figure 490](#) shows an example of communication with CTS flow control enabled.

Figure 490. RS232 CTS flow control



**Note:** For correct behavior, nCTS must be asserted 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 allows 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 de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] 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 de-assertion time equals
  - $(1 + (DEDT \times P)) \times f_{CK}$ , if  $P \neq 0$
  - $(1 + DEDT) \times f_{CK}$ , if  $P = 0$

where  $P = BRR[20:11]$

### 45.3.14 LPUART low-power management

The LPUART has advanced low-power mode functions allowing it to transfer properly data even when the `lpuart_pclk` clock is disabled.

The LPUART is able to wake up the MCU from low-power mode when the `UESM` bit is set. When the `usart_pclk` is gated, the LPUART 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

`lpuart_pclk` clock has to be activated to empty the LPUART data register.

In this case, the `lpuart_wkup` interrupt source is the `RXNE` set to '1'. The `RXNEIE` bit must be set before entering low-power mode.

- If FIFO mode is enabled

`lpuart_pclk` clock has to be activated

- to fill the `TXFIFO`
- or to empty the `RXFIFO`

In this case, the `lpuart_wkup` interrupt source can be:

- `RXFIFO` not empty. In this case, the `RXFNEIE` bit must be set before entering low-power mode.
- `RXFIFO` full. In this case, the `RXFFIE` bit must be set before entering low-power mode, the number of received data corresponds to the `RXFIFO` size, and the `RXFF` flag is not set .
- `TXFIFO` empty. In this case, the `TXFEIE` bit must be set before entering low-power mode.

This allows sending/receiving the data in the `TXFIFO/RXFIFO` during low-power mode.

To avoid overrun/underrun errors and transmit/receive data in low-power mode, the `lpuart_wkup` interrupt source can be one of the following events:

- `TXFIFO` threshold reached. In this case, the `TXFTIE` bit must be set before entering low-power mode.
- `RXFIFO` threshold reached. In this case, the `RXFTIE` bit must be set before entering low-power mode.

For example, the application can set the threshold to the maximum `RXFIFO` size if the 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 allows 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. In this case the **`lpuart_wkup`** interrupt is not mandatory for the wakeup. The `WUF` being set is sufficient to wakeup the MCU from low-power mode.

*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 will remain 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 *usart\_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 491](#) shows the behavior when the wakeup event is verified.



**Figure 491. Wakeup event verified (wakeup event = address match, FIFO disabled)**

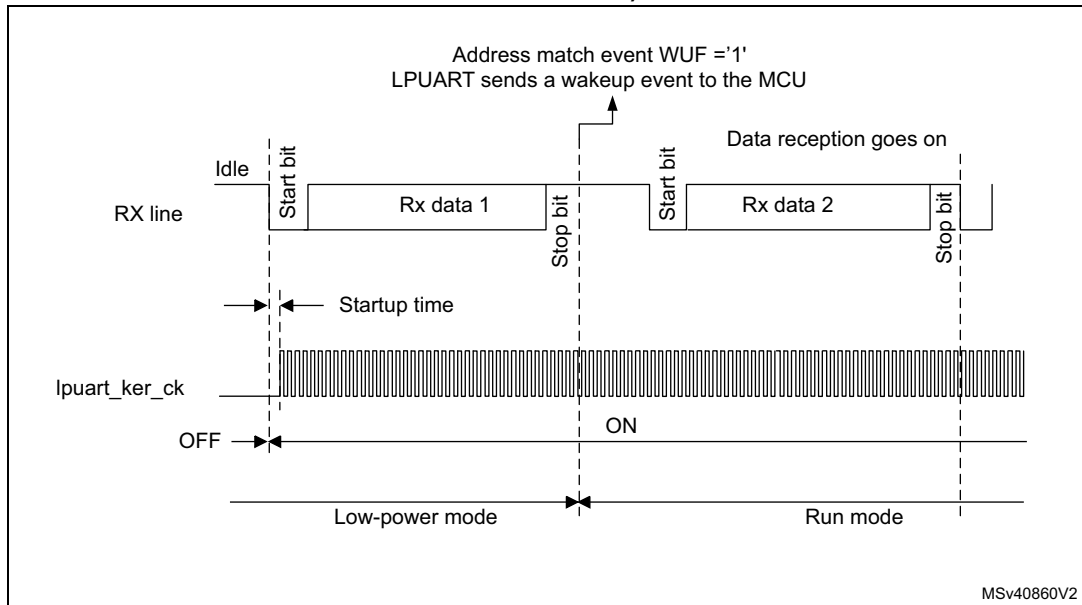
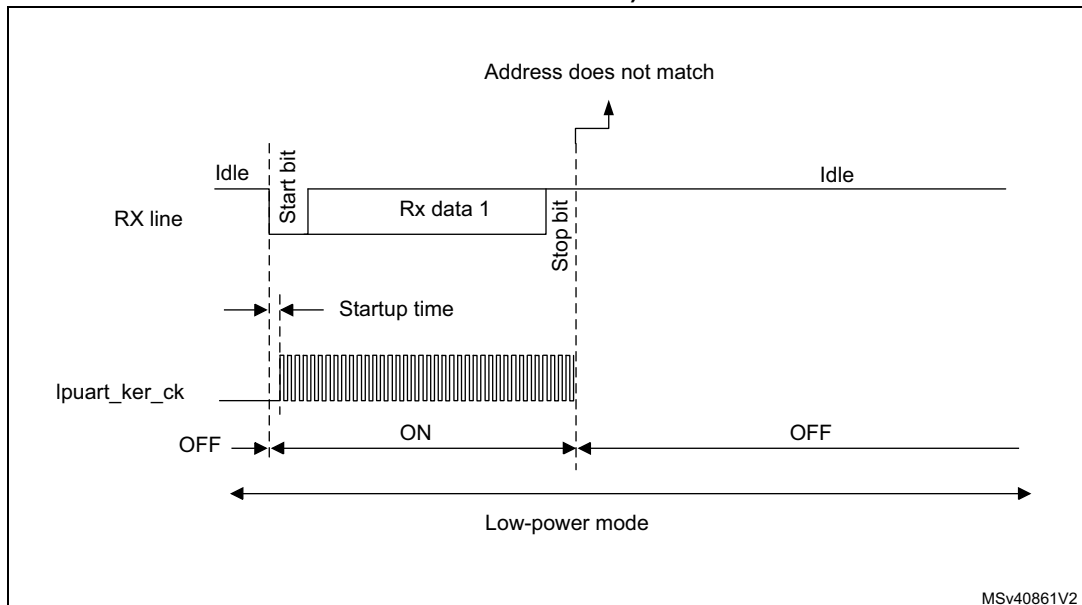


Figure 492 shows the behavior when the wakeup event is not verified.

**Figure 492. 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 allows to correctly wake up the MCU from low-power mode

The maximum baud rate allowing 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 45.3.8: 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 273: Tolerance of the LPUART receiver](#), the LPUART receiver tolerance equals 3.41%.

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

$$D_{WUmax} = t_{WULPUART} / (11 \times T_{bit\ Min})$$

$$T_{bit\ Min} = t_{WULPUART} / (11 \times D_{WUmax})$$

where  $t_{WULPUART}$  is the 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:

$$t_{WULPUART} = 3 \mu\text{s} \text{ (values provided only as examples; for correct values, refer to the device datasheet).}$$

$$D_{WUmax} = 3.41\% - 1\% = 2.41\%$$

$$T_{bit\ min} = 3 \mu\text{s} / (11 \times 2.41\%) = 11.32 \mu\text{s}.$$

As a result, the maximum baud rate allowing to wakeup correctly from low-power mode is:  $1/11.32 \mu\text{s} = 88.36 \text{ Kbaud}$ .

## 45.4 LPUART interrupts

Refer to [Table 269](#) for a detailed description of all LPUART interrupt requests.

**Table 275. LPUART interrupt requests**

Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Interrupt activated	
				lpuart_it	lpuart_wkup
Transmit data register empty	TXE	TXEIE	TXE cleared when a data is written in TDR	YES	NO
Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFNF cleared when TXFIFO is full.	YES	NO
Transmit FIFO Empty	TXFE	TXFEIE	TXFE cleared when the TXFIFO contains at least one data or by setting TXFRQ bit.	YES	YES
Transmit FIFO threshold reached	TXFT	TXFTIE	TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold	YES	YES

Table 275. LPUART interrupt requests (continued)

Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Interrupt activated	
				lpuart_it	lpuart_wkup
CTS interrupt	CTSIF	CTSIE	CTSIF cleared by software by setting CTSCF bit.	YES	NO
Transmission Complete	TC	TCIE	TC cleared when a data is written in TDR or by setting TCCF bit.	YES	NO
Receive data register not empty (data ready to be read)	RXNE	RXNEIE	RXNE cleared by reading RDR or by setting RXFRQ bit.	YES	YES
Receive FIFO Not Empty	RXFNE	RXFNEIE	RXFNE cleared when the RXFIFO is empty or by setting RXFRQ bit.	YES	YES
Receive FIFO Full	RXFF <sup>(1)</sup>	RXFFIE	RXFF cleared when the RXFIFO contains at least one data.	YES	YES
Receive FIFO threshold reached	RXFT	RXFTIE	RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold	YES	YES
Overrun error detected	ORE	RX-NEIE/RX-FNEIE	ORE cleared by setting ORECF bit.	YES	NO
Idle line detected	IDLE	IDLEIE	IDLE cleared by setting IDLECF bit.	YES	NO
Parity error	PE	PEIE	PE cleared by setting PECF bit.	YES	NO
Noise Flag, Overrun error and Framing Error in multibuffer communication.	NE or ORE or FE	EIE	NE cleared by setting NCF bit. ORE cleared by setting ORECF bit. FE flag cleared by setting FECF bit.	YES	NO
Character match	CMF	CMIE	CMF cleared by setting CMCF bit.	YES	NO
Wakeup from low-power mode	WUF	WUFIE	WUF is cleared by setting WUCF bit.	YES	YES
Transmit FIFO threshold reached	TXFT	TXFTIE	TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold	YES	YES
Receive FIFO threshold reached	RXFT	RXFTIE	RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold.	YES	YES

1. 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 will not be written and once n data are received and written in the RXFIFO, the RXFF interrupt will be asserted (RXFF flag is not set).

## 45.5 LPUART registers

Refer to [Section 1.2 on page 75](#) for a list of abbreviations used in register descriptions.

### 45.5.1 Control register 1 (LPUART\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

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	TXEIE/ TXFN FIE	TCIE	RXNEIE /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

*Note: When FIFO mode is disabled, this bit is reserved and must be kept at reset value.*

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

*Note: When FIFO mode is disabled, this bit is reserved and must be kept at reset value.*

Bit 29 **FIFOEN**:FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = '00': 1 Start bit, 8 Data bits, n Stop bit

M[1:0] = '01': 1 Start bit, 9 Data bits, n Stop bit

M[1:0] = '10': 1 Start bit, 7 Data bits, n Stop bit

This bit can only be written when the LPUART is disabled (UE=0).

*Note: In 7-bit data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.*

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart\_ker\_ck clock cycles. For more details, refer [Section 44.5.20: 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 45.3.13: 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 will be 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/TXFNFIE**: Transmit data register empty/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 **RXNEIE/RXFNEIE**: Receive data register not empty/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.*

### 45.5.2 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[7:4]								ADD[3:0]								Res.				Res.				Res.				Res.				MSBF1 RST				DATAINV				TXINV				RXINV																			
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				Res.				STOP[1:0]				Res.				Res.				Res.				Res.				Res.				Res.				Res.				Res.				Res.				Res.															
rw								rw				rw																																																			

Bits 31:28 **ADD[7:4]**: Address of the LPUART node

This bitfield gives the address of the LPUART node or a character code to be recognized. It is used to wake up the MCU with 7-bit address mark detection in multiprocessor communication during Mute mode or Stop mode. The MSB of the character sent by the transmitter should be equal to 1. It can also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on match. This bitfield can only be written when reception is disabled (RE = 0) or the LPUART is disabled (UE=0)

Bits 27:24 **ADD[3:0]**: Address of the LPUART node

This bitfield gives the address of the LPUART node or a character code to be recognized. This is used for wakeup with address mark detection in multiprocessor communication during Mute mode or low-power mode. This bitfield can only be written when reception is disabled (RE = 0) or the LPUART 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 ( $V_{DD} = 1/idle$ , Gnd=0/mark)  
1: TX pin signal values are inverted. ( $V_{DD} = 0/mark$ , Gnd=1/idle).  
This allows the use of an external inverter on the TX line.  
This 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 ( $V_{DD} = 1/idle$ , Gnd=0/mark)  
1: RX pin signal values are inverted. ( $V_{DD} = 0/mark$ , Gnd=1/idle).  
This allows the use of an external inverter on the RX line.  
This 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 allows 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.

### 45.5.3 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
RXFTCFG		RXFTIE	RXFTCFG		Res.	TXFTIE	WUFIE	WUS[2:0]		Res.	Res.	Res.	Res.		
rw		rw	rw			rw	rw	rw	rw						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HD SEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

- Bits 31:29 **TXFTCFG**: 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.
- Bit28 **RXFTE**: 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**: 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 44.4: 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 44.4: 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 allows the user to activate the external transceiver control, through the DE signal.  
0: DE function is disabled.  
1: DE function is enabled. The DE signal is output on the RTS pin.  
This bit can only be written when the 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 will be 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 allows 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 nCTS input is asserted (tied to 0). If the nCTS input is deasserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.  
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 nRTS output is asserted (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.

### 45.5.4 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]**

*Note:* It is forbidden to write values lower than 0x300 in the LPUART\_BRR register.  
 Provided that LPUART\_BRR must be ≥ 0x300 and LPUART\_BRR is 20 bits, a care should be taken when generating high baud rates using high fck values. fck must be in the range [3 x baud rate..4096 x baud rate].

### 45.5.5 Request register (LPUART\_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
											w	w	w	w	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This will set 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 allows 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.

### 45.5.6 Interrupt and status register (LPUART\_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 00 if FIFO is disabled

XX = 08 if FIFO is enabled and Smartcard mode is disabled

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.	TXE/TX FNF	TC	RXNE/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: No break character is transmitted  
1: Break character will be transmitted
- 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 nCTS input pin.  
0: nCTS line set  
1: nCTS line reset  
*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*
- Bit 9 **CTSIF**: CTS interrupt flag  
This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART\_ICR register.  
An interrupt is generated if CTSIE=1 in the LPUART\_CR3 register.  
0: No change occurred on the nCTS status line  
1: A change occurred on the nCTS status line  
*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*
- Bit 8 Reserved, must be kept at reset value.

**Bit 7 TXE/TXFNF:** Transmit data register empty/TXFIFO not full

When FIFO mode is disabled, 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.

When FIFO mode is enabled, 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.

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 will be set at the same time).

An interrupt is generated if the TXEIE/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 TXE/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 will be set immediately.

**Bit 5 RXNE/RXFNE:** Read data register not empty/RXFIFO 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 a read to the LPUART\_RDR register. The

RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART\_RQR register.

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 RXNE/RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART\_RQR register.

An interrupt is generated if RXNEIE/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 will not be set again until the RXNE bit has been set (i.e. a new idle line occurs).

If Mute mode is enabled (MME=1), IDLE is set if the 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 (RXFF = 1 in case FIFO mode is enabled). It is cleared by a software, writing 1 to the ORECF, in the LPUART\_ICR register.

An interrupt is generated if RXNEIE/ RXFNEIE=1 or EIE = 1 in the LPUART\_CR1 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 NFCF 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/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.*

*In FIFO mode, 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\_CR1 register.

- 0: No Framing error is detected
- 1: Framing error or break character is detected

*Note: In FIFO mode, 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: In FIFO mode, this error is associated with the character in the LPUART\_RDR.*

### 45.5.7 Interrupt flag clear register (LPUART\_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
						w			w		w	w	w	w	w



Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: 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 44.4: 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 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.

### 45.5.8 Receive data register (LPUART\_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value

Contains the received data character.

The RDR register provides the parallel interface between the input shift register and the internal bus (see [Figure 479](#)).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

### 45.5.9 Transmit data register (LPUART\_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see [Figure 479](#)).

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.*

### 45.5.10 Prescaler register (LPUART\_PRESC)

This register can only be written when the LPUART is disabled (UE=0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The LPUART input clock can be divided by a prescaler:

0000: input clock not divided

0001: input clock divided by 2

0010: input clock divided by 4

0011: input clock divided by 6

0100: input clock divided by 8

0101: input clock divided by 10

0110: input clock divided by 12

0111: input clock divided by 16

1000: input clock divided by 32

1001: input clock divided by 64

1010: input clock divided by 128

1011: input clock divided by 256

Remaining combinations: Reserved.

*Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value will be «1011» i.e. input clock divided by 256.*

### 45.5.11 LPUART register map

The table below gives the LPUART register map and reset values.

**Table 276. LPUART 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	LPUART_CR1	RXFIE	TXFIE	FIFOEN	M1	Res.	Res.	DEAT4	DEAT3	DEAT2	DEAT1	DEAT0	DEDT4	DEDT3	DEDT2	DEDT1	DEDT0	Res.	CMIE	MME	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE	
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x04	LPUART_CR2	ADD[7:4]				ADD[3:0]				Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	Res.	STOP	[1:0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	LPUART_CR3	TXFT CFG		RXFIE	RXFT CFG		Res.	TXFTIE	WUFIE	WUS	[1:0]	Res.	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	Res.	Res.	Res.	EIE		
	Reset value	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0C	LPUART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:0]																					
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x10-0x14	Reserved																																	
0x18	LPUART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																	
0x1C	LPUART_ISR	Res.	Res.	Res.	Res.	TXFT	RXFT	RXFF	TXFF	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE/TXFNF	TC	RXNE	IDLE	ORE	NE	FE	PE	
	Reset value					0	0	0	0	0	0	0	0	0	0	0							0	0	0	1	0	0	0	0	0	0	0	0
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WJCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value												0											0										
0x24	LPUART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																									0	0	0	0	0	0	0	0	0
0x28	LPUART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																									0	0	0	0	0	0	0	0	0
0x2C	LPUART_PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																	0

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.



## 46 Serial peripheral interface (SPI)

### 46.1 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.

### 46.2 SPI main features

- Master or slave operation
- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- 4-bit to 16-bit data size selection
- Multimaster mode capability
- 8 master mode baud rate prescalers up to  $f_{PCLK}/2$
- Slave mode frequency up to  $f_{PCLK}/2$ .
- NSS management by hardware or software for both master and slave: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- SPI Motorola support
- Hardware CRC feature for reliable communication:
  - CRC value can be transmitted as last byte in Tx mode
  - Automatic CRC error checking for last received byte
- Master mode fault, overrun flags with interrupt capability
- CRC Error flag
- Two 32-bit embedded Rx and Tx FIFOs with DMA capability
- SPI TI mode support

### 46.3 SPI implementation

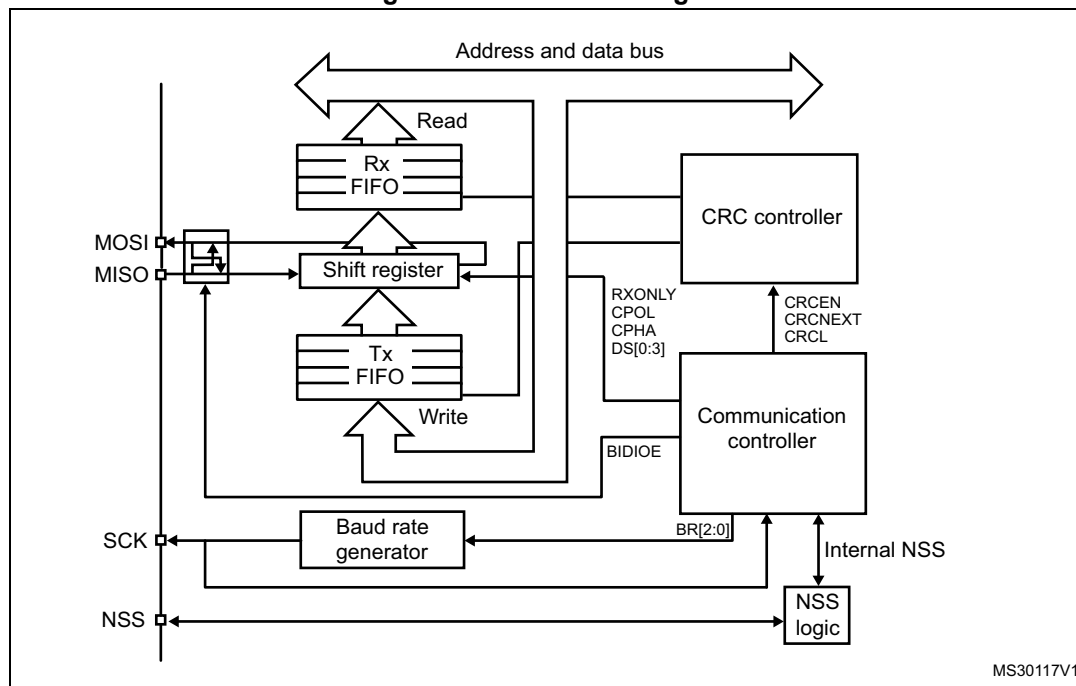
This manual describes the SPI/I2S implementation in STM32L4Rxxx and STM32L4Sxxx devices.

## 46.4 SPI functional description

### 46.4.1 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 493](#).

Figure 493. 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 46.4.5: 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.

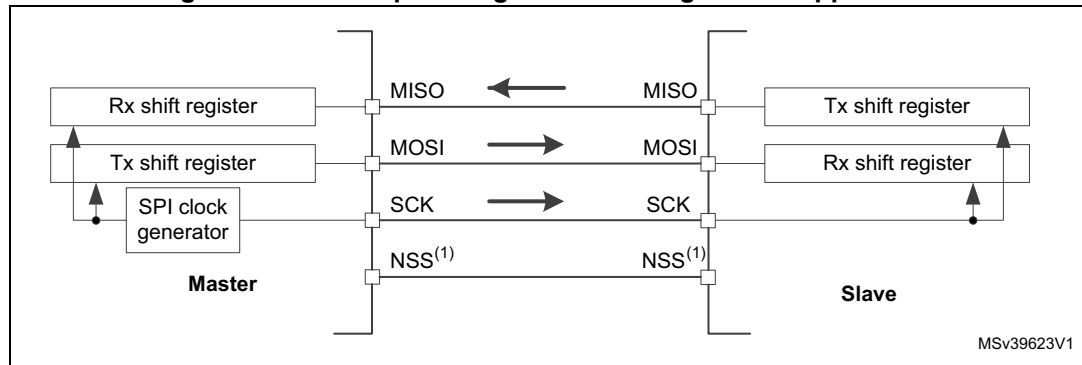
## 46.4.2 Communications between one master and one slave

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 494. Full-duplex single master/ single slave application



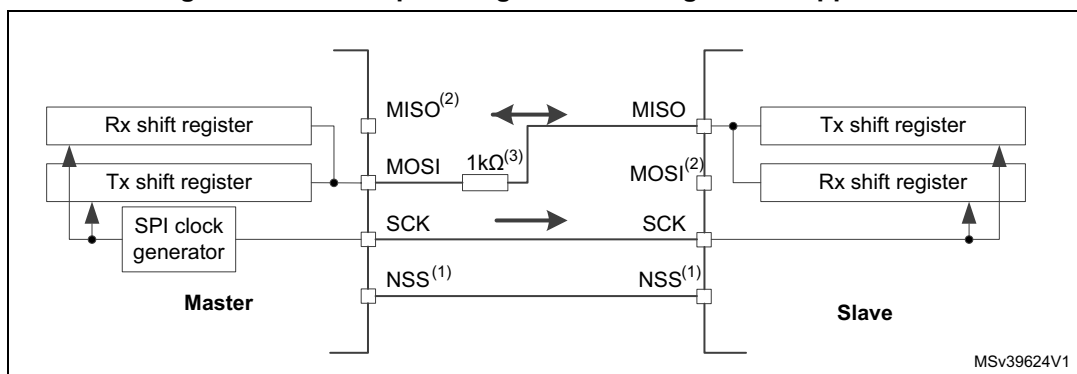
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see [Section 46.4.5: Slave select \(NSS\) pin management](#).

### 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 495. Half-duplex single master/ single slave application



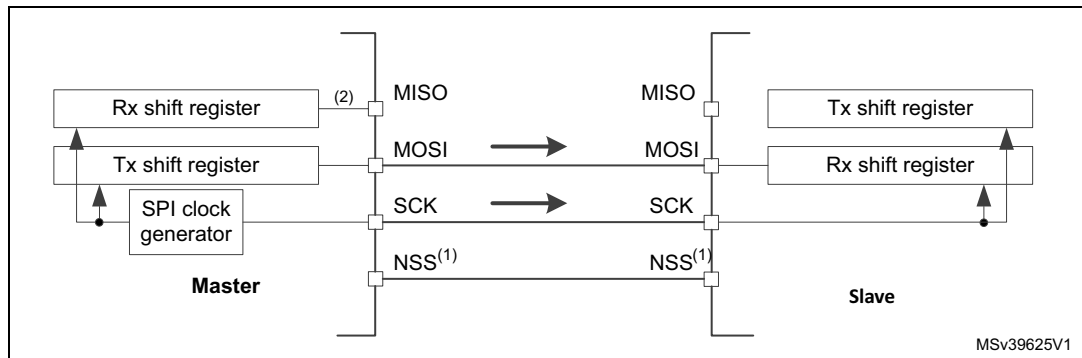
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see [Section 46.4.5: Slave select \(NSS\) pin management](#).
2. In 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 blowing between them at this situation.

### Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the RXONLY bit in the SPIx\_CR2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO and MOSI pins pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode (RXONLY=0):** The configuration settings are the same as for full-duplex. The application has to ignore the information captured on the unused input pin. This pin can be used as a standard GPIO.
- **Receive-only mode (RXONLY=1):** The application can disable the SPI output function by setting the RXONLY bit. In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see [46.4.5: 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 496. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode)**



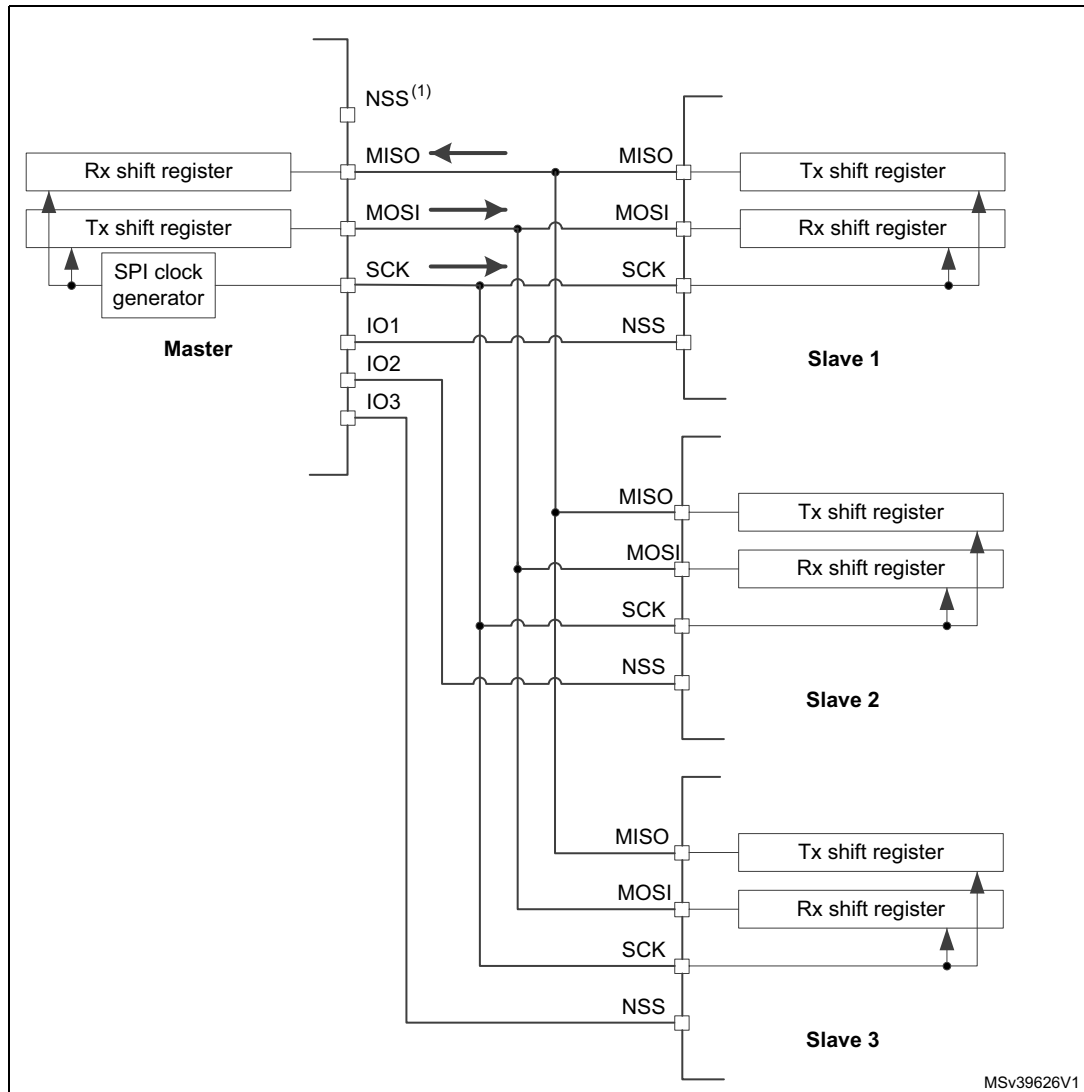
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see [Section 46.4.5: Slave select \(NSS\) pin management](#).
2. An 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. OVF flag).
3. In this configuration, both the MISO pins can be used as GPIOs.

**Note:** *Any simplex communication can be alternatively replaced by a variant of the half-duplex communication with a constant setting of the transaction direction (bidirectional mode is enabled while BDIO bit is not changed).*

### 46.4.3 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 497](#)). 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 497. 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 [Section 8.3.7: I/O alternate function input/output on page 312](#)).

#### 46.4.4 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.

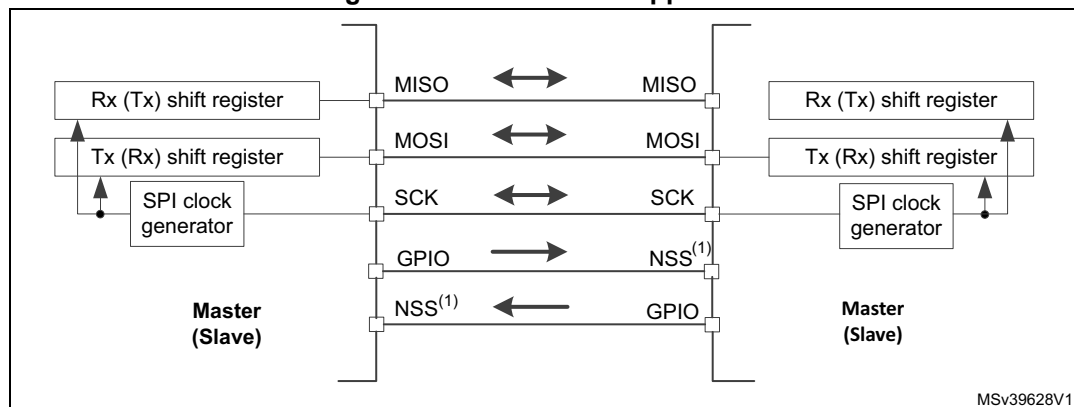
The connection of more than two SPI nodes working at this mode is impossible as only one node can apply its output on a common data line at time.

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 498. 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.

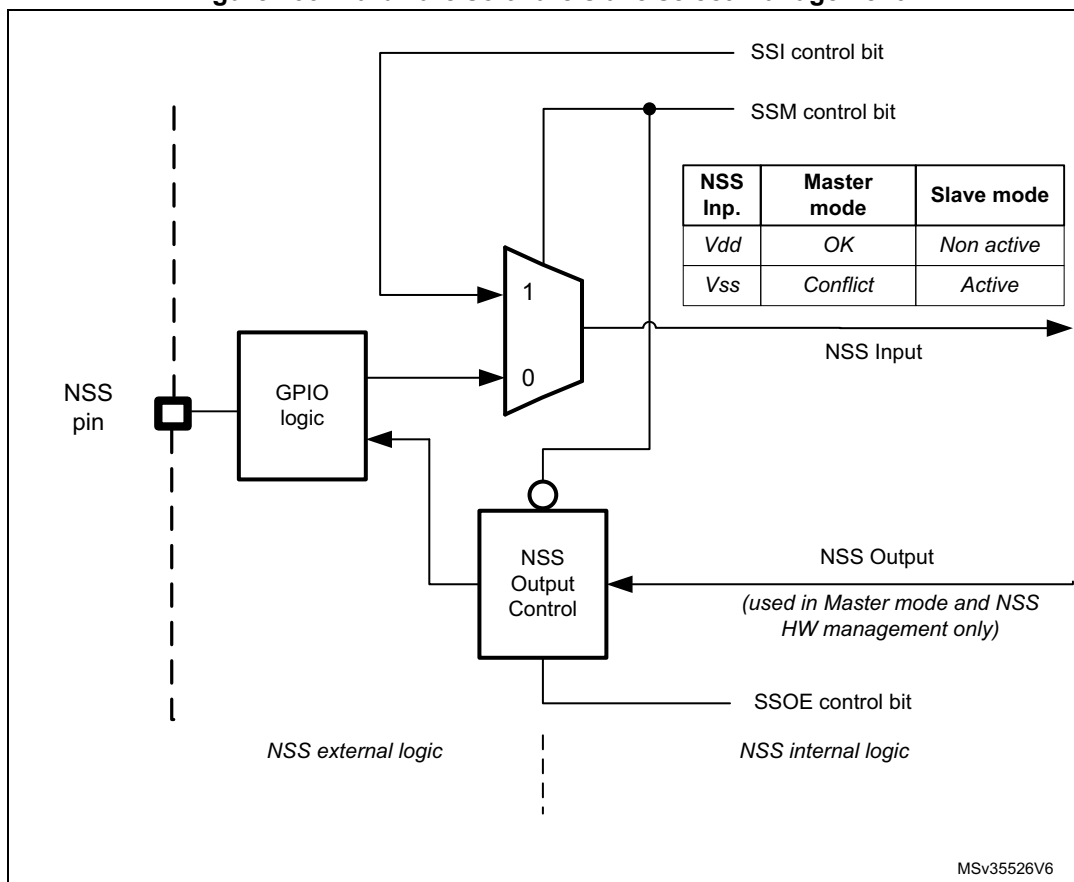
#### 46.4.5 Slave select (NSS) pin management

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 499. Hardware/software slave select management



### 46.4.6 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.

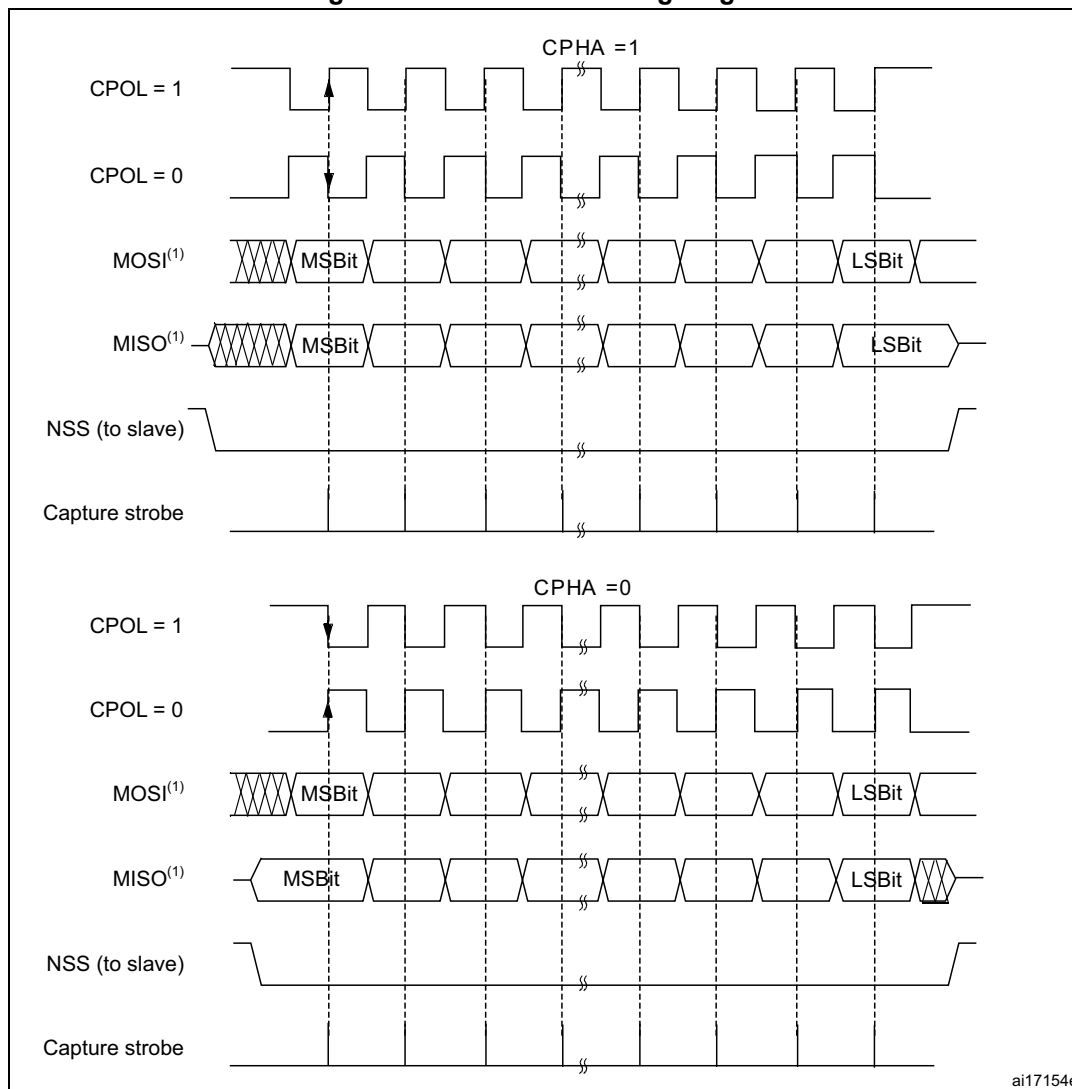
If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures 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 500, 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).

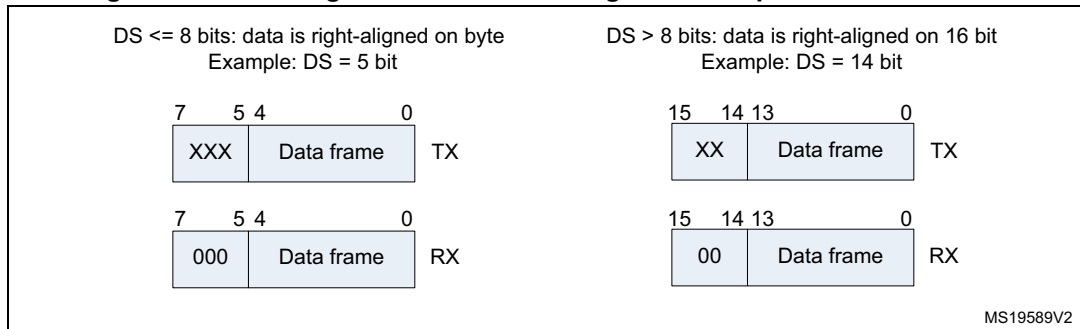
Figure 500. Data clock timing diagram



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 501). During communication, only bits within the data frame are clocked and transferred.

**Figure 501. 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.

#### 46.4.7 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 can't be set at the same time).
  - d) Configure the LSBFIRST bit to define the frame format (Note: 2).
  - e) Configure the CRCL and CRCEN bits if CRC is needed (while SCK clock signal is at idle state).
  - f) Configure SSM and SSI (Notes: 2 & 3).
  - g) Configure the MSTR bit (in multimaster NSS configuration, avoid conflict state on NSS if master is configured to prevent MODF error).
3. Write to SPI\_CR2 register:
  - a) Configure the DS[3:0] bits to select the data length for the transfer.
  - b) Configure SSOE (Notes: 1 & 2 & 3).
  - c) Set the FRF bit if the TI protocol is required (keep NSSP bit cleared in TI mode).
  - d) Set the NSSP bit if the NSS pulse mode between two data units is required (keep CHPA and TI bits cleared in NSSP mode).
  - e) Configure the FRXTH bit. The RXFIFO threshold must be aligned to the read access size for the SPIx\_DR register.
  - f) Initialize LDMA\_TX and LDMA\_RX bits if DMA is used in packed mode.
4. Write to SPI\_CRCPR register: Configure the CRC polynomial if needed.
5. Write proper DMA registers: Configure DMA streams dedicated for SPI Tx and Rx in DMA registers if the DMA streams are used.

- Note:
- (1) Step is not required in slave mode.
  - (2) Step is not required in TI mode.
  - (3) Step is not required in NSSP mode.
  - (4) The step is not required in slave mode except slave working at TI mode

#### 46.4.8 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.

#### 46.4.9 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 46.4.14: 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 46.4.13: 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 503](#) through [Figure 506](#).



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 46.4.10: 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 multislave system to select just one of the slaves for communication. In a single slave system it is not necessary to control the slave with NSS, but it is often better to provide the pulse here too, to synchronize the slave with the beginning of each data sequence. NSS can be managed by both software and hardware (see [Section 46.4.5: 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.

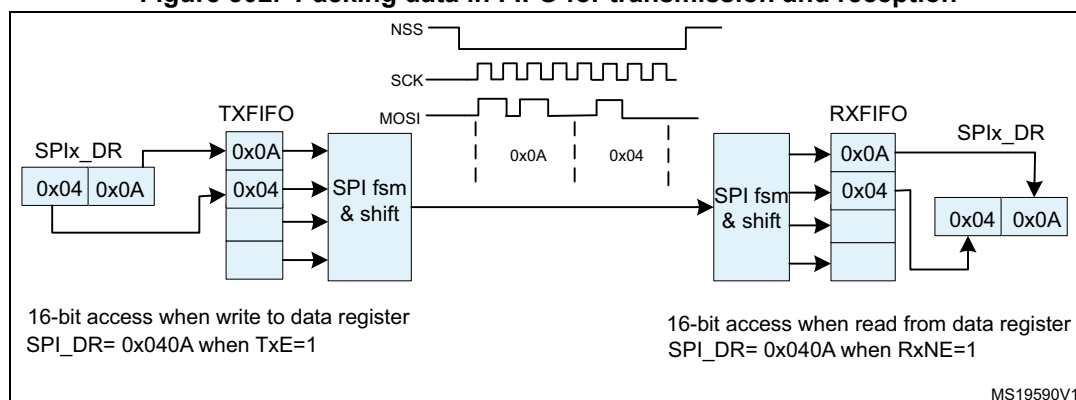
### Data packing

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 502](#) 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 502. Packing data in FIFO for transmission and reception**



**Communication using DMA (direct memory addressing)**

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXE or RXNE enable bit in the SPIx\_CR2 register is set. Separate requests must be issued to the Tx and Rx buffers.

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes to the SPIx\_DR register.
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPIx\_DR register.

See [Figure 503](#) through [Figure 506](#).

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 1652](#).)

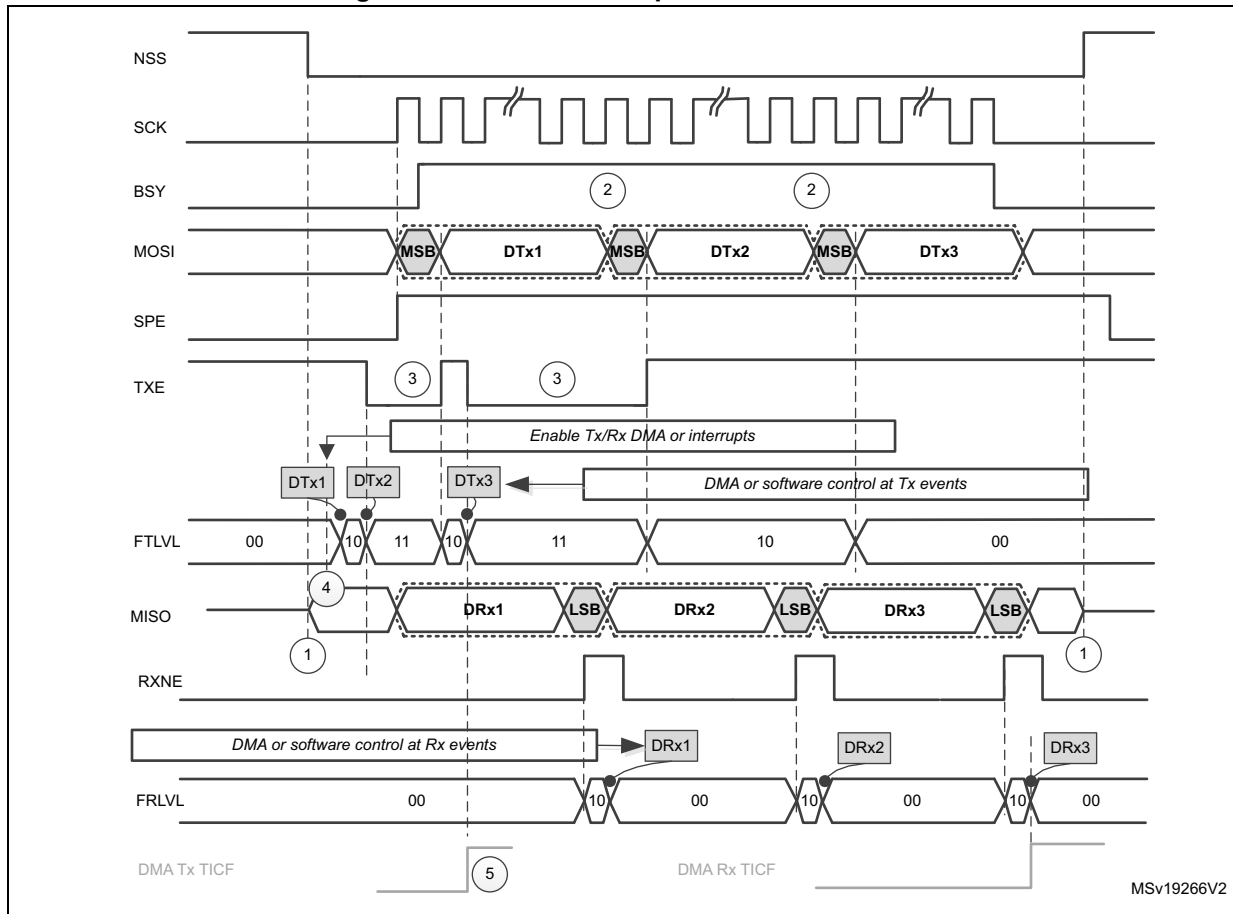
## Communication diagrams

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 503 on page 1656](#) through [Figure 506 on page 1659](#).

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 503. Master full-duplex communication



Assumptions for master full-duplex communication example:

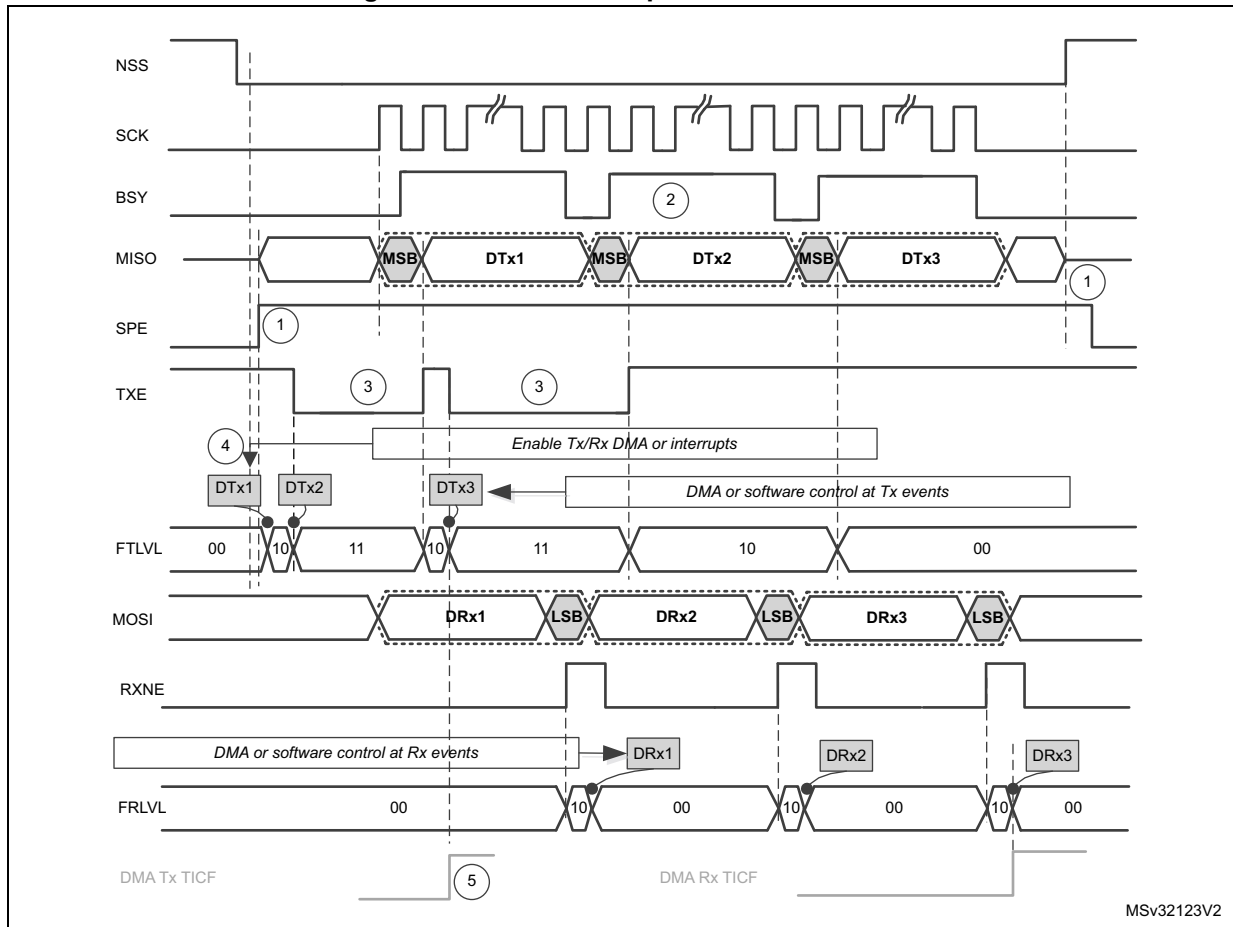
- Data size > 8 bit

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 1655](#) for details about common assumptions and notes.

Figure 504. Slave full-duplex communication



Assumptions for slave full-duplex communication example:

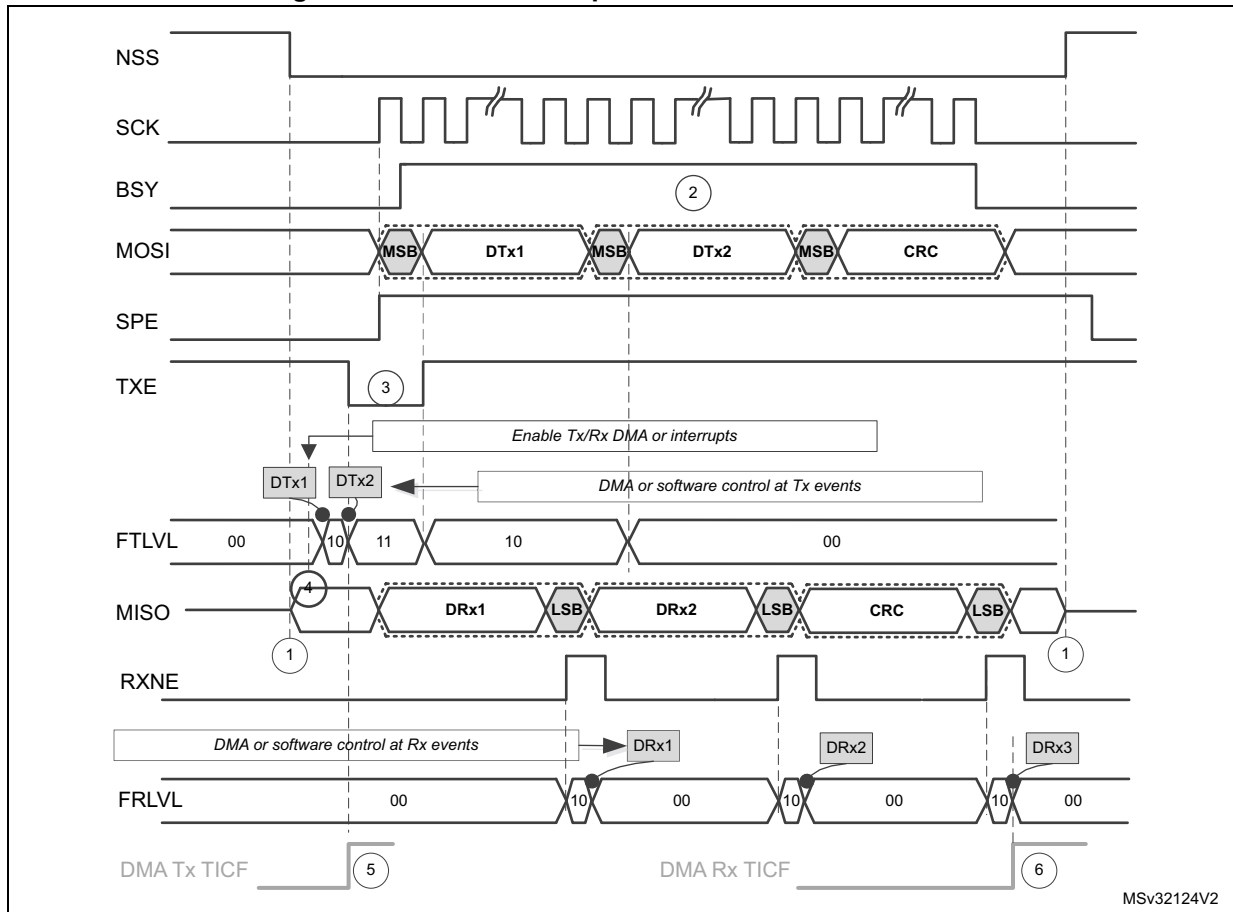
- Data size > 8 bit

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 1655](#) for details about common assumptions and notes.

Figure 505. Master full-duplex communication with CRC



Assumptions for master full-duplex communication with CRC example:

- Data size = 16 bit
- CRC enabled

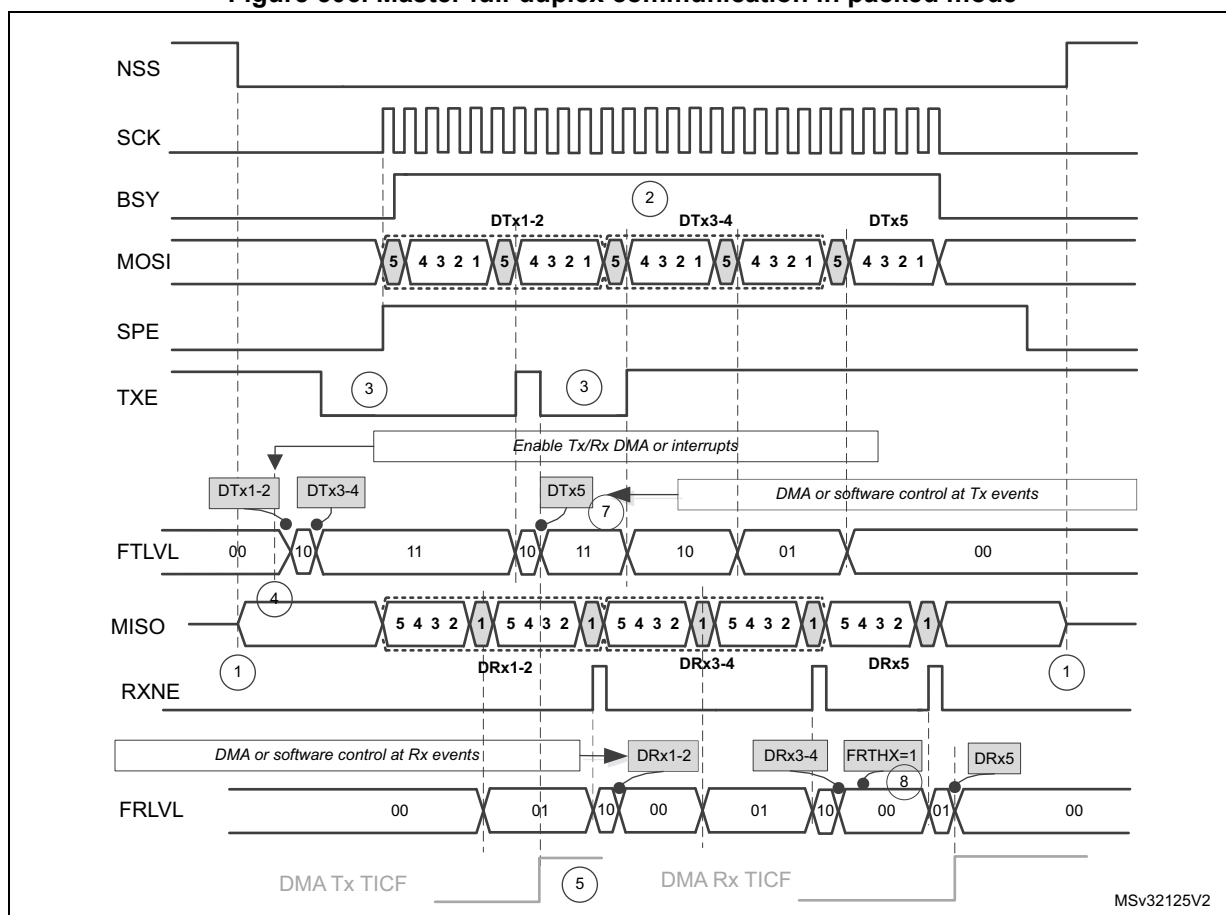
If DMA is used:

- 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 1655](#) for details about common assumptions and notes.



Figure 506. 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 1655](#) for details about common assumptions and notes.

### 46.4.10 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.

### 46.4.11 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 46.4.14: 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.

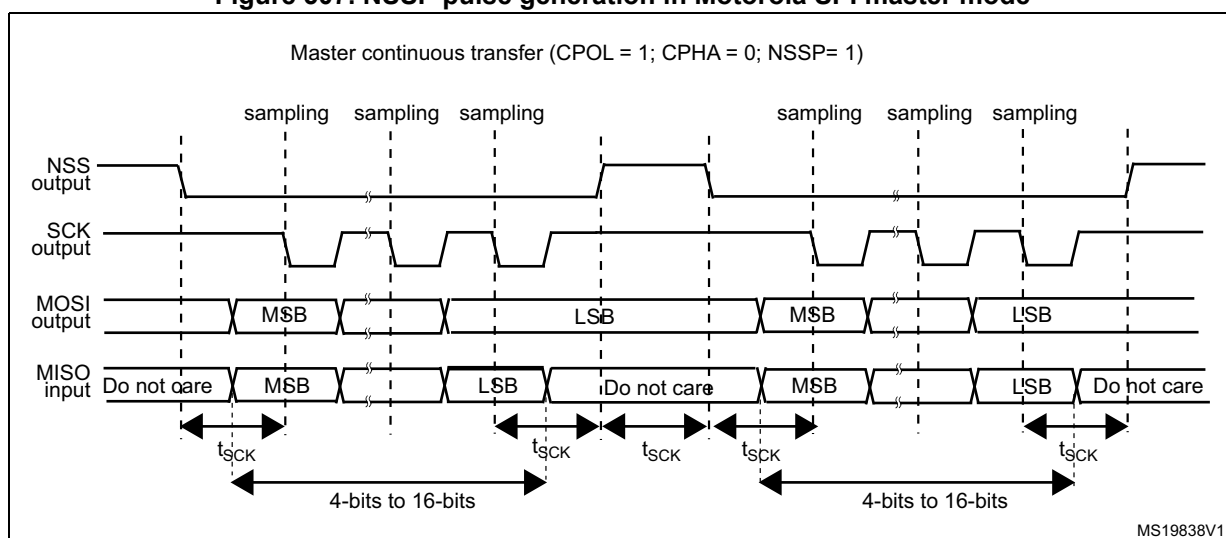
The FRE flag is cleared when SPIx\_SR register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no longer guaranteed and communications should be reinitiated by the master when the slave SPI is enabled again.

### 46.4.12 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 507 illustrates NSS pin management when NSSP pulse mode is enabled.

Figure 507. 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.

### 46.4.13 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 508). Any baud rate can be used, making it possible to determine this moment with optimal flexibility. However, the baud rate is generally set to the external master clock baud rate. The delay for the MISO signal to become HiZ ( $t_{release}$ ) depends on internal resynchronization and on the

baud rate value set in through the BR[2:0] bits in the SPIx\_CR1 register. It is given by the formula:

$$\frac{t_{\text{baud\_rate}}}{2} + 4 \times t_{\text{pclk}} < t_{\text{release}} < \frac{t_{\text{baud\_rate}}}{2} + 6 \times t_{\text{pclk}}$$

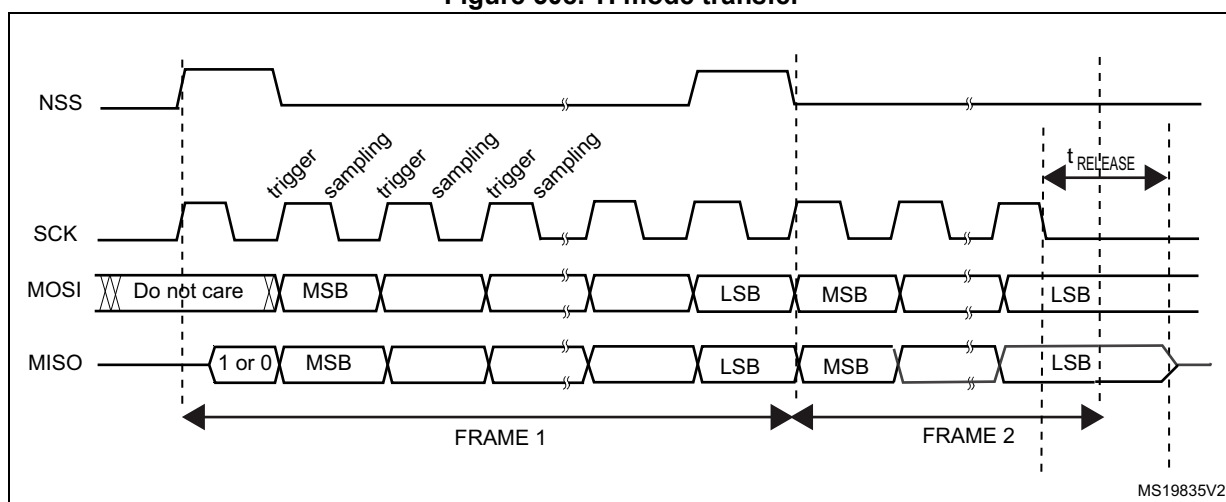
If the slave detects a misplaced NSS pulse during a data frame transaction the TIFRE flag is set.

If the data size is equal to 4-bits or 5-bits, the master in full-duplex mode or transmit-only mode uses a protocol with one more dummy data bit added after LSB. TI NSS pulse is generated above this dummy bit clock cycle instead of the LSB in each period.

This feature is not available for Motorola SPI communications (FRF bit set to 0).

*Figure 508: TI mode transfer* shows the SPI communication waveforms when TI mode is selected.

**Figure 508. TI mode transfer**



#### 46.4.14 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 will follow after the transaction of the currently processed data frame. The CRCNEXT bit must be set before the end of the last data frame transaction. CRC calculation is frozen during CRC transaction.

The received CRC is stored in the RXFIFO like a data byte or word. That is why in CRC mode only, the reception buffer has to be considered as a single 16-bit buffer used to receive only one data frame at a time.

A CRC-format transaction usually takes one more data frame to communicate at the end of data sequence. However, when setting an 8-bit data frame checked by 16-bit CRC, two more frames are necessary to send the complete CRC.

When the last CRC data is received, an automatic check is performed comparing the received value and the value in the SPIx\_RXCRC register. Software has to check the CRCERR flag in the SPIx\_SR register to determine if the data transfers were corrupted or not. Software clears the CRCERR flag by writing '0' to it.

After the CRC reception, the CRC value is stored in the RXFIFO and must be read in the SPIx\_DR register in order to clear the RXNE flag.

### **CRC transfer managed by DMA**

When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication is automatic (with the exception of reading CRC data in receive only mode). The CRCNEXT bit does not have to be handled by the software. The counter for the SPI transmission DMA channel has to be set to the number of data frames to transmit excluding the CRC frame. On the receiver side, the received CRC value is handled automatically by DMA at the end of the transaction but user must take care to flush out received CRC information from RXFIFO as it is always loaded into it. In full-duplex mode, the counter of the reception DMA channel can be set to the number of data frames to receive including the CRC, which means, for example, in the specific case of an 8-bit data frame checked by 16-bit CRC:

$$\text{DMA\_RX} = \text{Numb\_of\_data} + 2$$

In receive only mode, the DMA reception channel counter should contain only the amount of data transferred, excluding the CRC calculation. Then based on the complete transfer from DMA, all the CRC values must be read back by software from FIFO as it works as a single buffer in this mode.

At the end of the data and CRC transfers, the CRCERR flag in the SPIx\_SR register is set if corruption occurred during the transfer.

If packing mode is used, the LDMA\_RX bit needs managing if the number of data is odd.

### **Resetting the SPIx\_TXCRC and SPIx\_RXCRC values**

The SPIx\_TXCRC and SPIx\_RXCRC values are cleared automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode (not available in receive-only mode) in order to transfer data without any interruption, (several data blocks covered by intermediate CRC checking phases).

If the SPI is disabled during a communication the following sequence must be followed:

1. Disable the SPI
2. Clear the CRCEN bit
3. Enable the CRCEN bit
4. Enable the SPI

*Note: When the SPI 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 can't be used at NSS Pulse mode when NSS hardware mode should be applied at slave normally (see more details at the product errata sheet).*

*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.*

## 46.5 SPI interrupts

During SPI communication an interrupts can be generated by the following events:

- Transmit TXFIFO ready to be loaded
- Data received in Receive RXFIFO
- Master mode fault
- Overrun error
- TI frame format error
- CRC protocol error

Interrupts can be enabled and disabled separately.

**Table 277. 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	

## 46.6 SPI registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit). SPI\_DR in addition by can be accessed by 8-bit access.

### 46.6.1 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	BIDI OE	CRC EN	CRC NEXT	CRCL	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bit 15 **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 **CRCEEN**: 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 1651](#).*

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.*

Bit1 **CPOL**: Clock polarity  
 0: CK to 0 when idle  
 1: CK to 1 when idle

*Note: This bit should not be changed when communication is ongoing.  
 This bit is not used in SPI TI mode 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.*

### 46.6.2 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

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 1651](#) 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 1651](#) 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

### 46.6.3 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[2:0]		FRE	BSY	OVR	MODF	CRC ERR	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 46.4.11: 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 46.4.10: SPI status flags and Procedure for disabling the SPI on page 1651](#).*
- 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 1661](#) 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
  - 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

### 46.6.4 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 Tx FIFO (See [Section 46.4.9: 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.*

### 46.6.5 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 (0007h) is the reset value of this register. Another polynomial can be configured as required.

*Note: The polynomial value should be odd only. No even value is supported.*

### 46.6.6 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.*

### 46.6.7 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.*

### 46.6.8 SPI register map

Table 278 shows the SPI register map and reset values.

**Table 278. SPI register map and reset values**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPIx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BIDIMODE	BIDIOE	CRCEN	CRCNEXT	CRCL	RXONLY	SSM	SSI	LSBFIRST	SPE	BR [2:0]		MSTR	CPOL	CPHA	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	SPIx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDMA_TX	LDMA_RX	FRXTH	DS[3:0]			TXEIE	RXNEIE	ERRIE	FRF	NSSP	SSOE	TXDMAEN	RXDMAEN	
	Reset value																		0	0	0	0	1	1	1	0	0	0	0	0	0	0	
0x08	SPIx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTLVL[1:0]	FRLVL[1:0]		FRE	BSY	OVR	MODF	CRCERR	Res.	TXE	RXNE		
	Reset value																				0	0	0	0	0	0	0	0	0	0	1	0	
0x0C	SPIx_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DR[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x10	SPIx_CRCPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCPCPOLY[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	SPIx_RXCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RxCRC[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SPIx_TXCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TxCRC[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to Section 2.2.2 on page 82 for the register boundary addresses.



## 47 Serial audio interface (SAI)

### 47.1 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 sub-blocks. Each block has its own clock generator and I/O line controller.

The SAI can work in master or slave configuration. The audio sub-blocks can be either receiver or transmitter and can work synchronously or not (with respect to the other one).

The SAI can be connected with other SAIs to work synchronously.

## 47.2 SAI main features

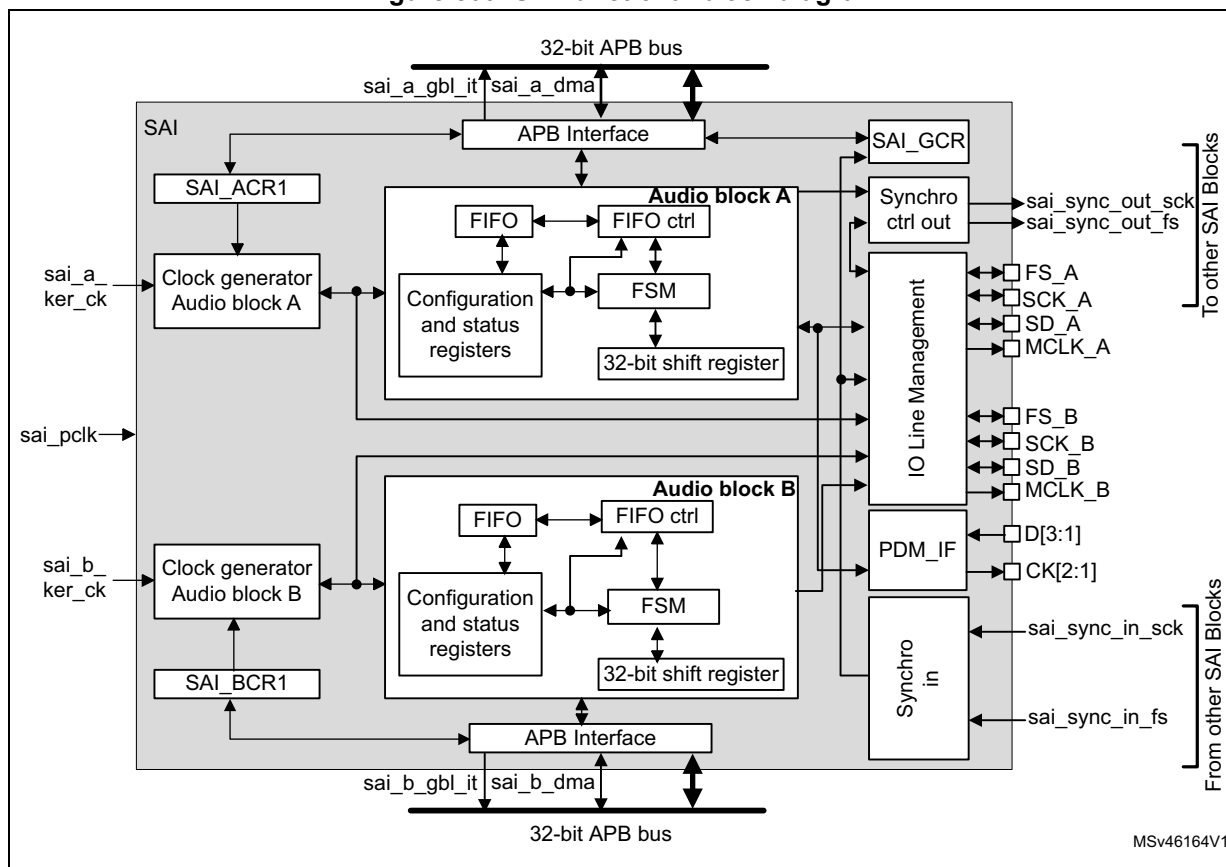
- Two independent audio sub-blocks which can be transmitters or receivers with their respective FIFO.
- 8-word integrated FIFOs for each audio sub-block.
- Synchronous or asynchronous mode between the audio sub-blocks.
- Possible synchronization between multiple SAIs.
- Master or slave configuration independent for both audio sub-blocks.
- Clock generator for each audio block to target independent audio frequency sampling when both audio sub-blocks 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.
- Interruption sources when enabled:
  - Errors,
  - FIFO requests.
- 2-channel DMA interface.

## 47.3 SAI functional description

### 47.3.1 SAI block diagram

Figure 509 shows the SAI block diagram while Table 279 and Table 280 list SAI internal and external signals.

Figure 509. SAI functional block diagram



The SAI is mainly composed of two audio sub-blocks 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 sub-blocks 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 sub-block 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 will be 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 sub-block.

### 47.3.2 SAI pins and internal signals

**Table 279. 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_sync_out_sck, sai_sync_out_fs	Output	Internal clock and frame synchronization output signals exchanged with other SAI blocks.
sai_sync_in_sck, sai_sync_in_fs	Input	Internal clock and frame synchronization input signals exchanged with other SAI blocks.
sai_a_ker_ck/ sai_b_ker_ck	Input	Audio block A/B kernel clock.
sai_pclk	Input	APB clock.

**Table 280. 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.
SAI_D[3:1]	Input	PDM bitstream data.

### 47.3.3 Main SAI modes

Each audio sub-block 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 sub-block is configured in asynchronous mode, then SCK\_x and FS\_x pins are configured as inputs.
- If the SAI sub-block is configured to operate synchronously with another SAI interface or with the second audio sub-block, 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 sub-block 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 will be 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.

## 47.3.4 SAI synchronization mode

There are two levels of synchronization, either at audio sub-block level or at SAI level.

### Internal synchronization

An audio sub-block can be configured to operate synchronously with the second audio sub-block 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 and 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.

### External synchronization

The audio sub-blocks can also be configured to operate synchronously with another SAI. This can be done as follow:

1. The SAI, which is configured as the source from which the other SAI is synchronized, has to define which of its audio sub-block is supposed to provide the FS and SCK signals to other SAI. This is done by programming SYNCOUT[1:0] bits.
2. The SAI which shall receive the synchronization signals has to select which SAI will provide the synchronization by setting the proper value on SYNCIN[1:0] bits. For each of the two SAI audio sub-blocks, the user must then specify if it operates synchronously with the other SAI via the SYNCEN bit.

*Note:* SYNCIN[1:0] and SYNCOUT[1:0] bits are located into the SAI\_GCR register, and SYNCEN bits into SAI\_xCR1 register.

If both audio sub-blocks in a given SAI need to be synchronized with another SAI, it is possible to choose one of the following configurations:

- Configure each audio block to be synchronous with another SAI block through the SYNCEN[1:0] bits.
- Configure one audio block to be synchronous with another SAI through the SYNCEN[1:0] bits. The other audio block is then configured as synchronous with the second SAI audio block through SYNCEN[1:0] bits.

The following table shows how to select the proper synchronization signal depending on the SAI block used. For example SAI2 can select the synchronization from SAI1 by setting SAI2 SYNCIN to 0. If SAI1 wants to select the synchronization coming from SAI2, SAI1 SYNCIN must be set to 1. Positions noted as 'Res.' shall not be used.

**Table 281. External synchronization selection**

Block instance	SYNCIN= 3	SYNCIN= 2	SYNCIN= 1	SYNCIN= 0
SAI1	Res.	Res.	SAI2 sync	Res.
SAI2	Res.	Res.	Res.	SAI1 sync

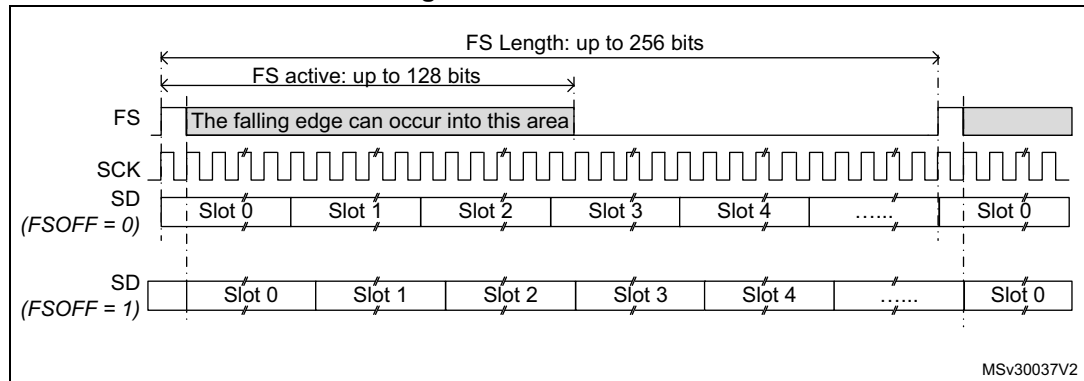
### 47.3.5 Audio data size

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.

### 47.3.6 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 510](#) illustrates this flexibility.

**Figure 510. Audio frame**



In AC'97 mode or in SPDIF mode (bit PRTCFCG[1:0] = 10 or PRTCFCG[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 will be extended to 0 or the SD line will be 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 NOMCK 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 NOMCK is set, the (FRL+1) field can take any value from 8 to 256. Refer to [Section 47.3.8: 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 on-going audio frame. In this case an error will be generated. For more details refer to [Section 47.3.14: 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 47.3.14: 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 will be managed as described in [Section 47.3.14: Error flags](#), but the audio communication flow will not be interrupted.

### Frame synchronization active level length

The FSALL[6:0] bits of the SAI\_xFRCR register allow configuring 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 mode.

### 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 allows 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 will have:

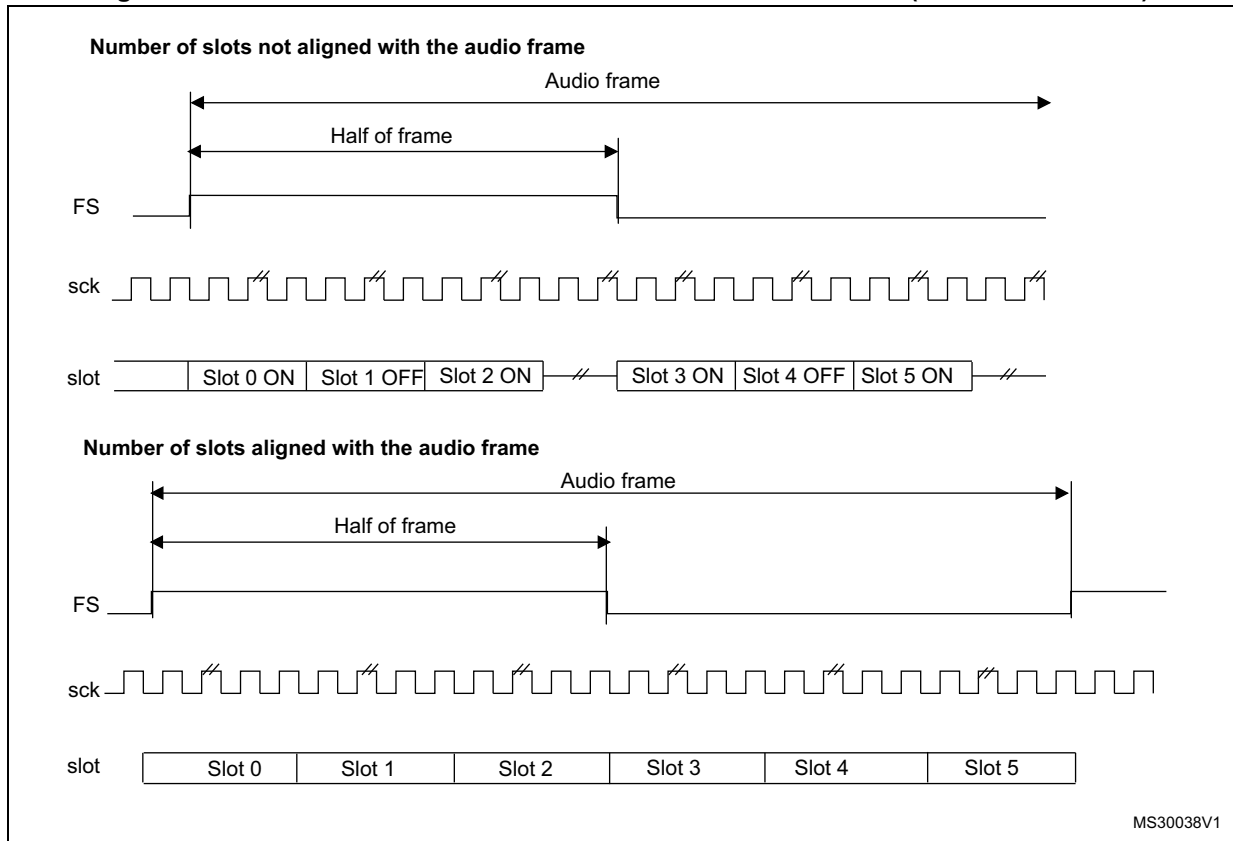
- 0: start of frame, like for instance the PCM/DSP, TDM, AC'97, audio protocols,
- 1: start of frame and channel side identification within the audio frame like for the I2S, the MSB or LSB-justified protocols.

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 511. FS role is start of frame + channel side identification (FSDEF = TRIS = 1)**

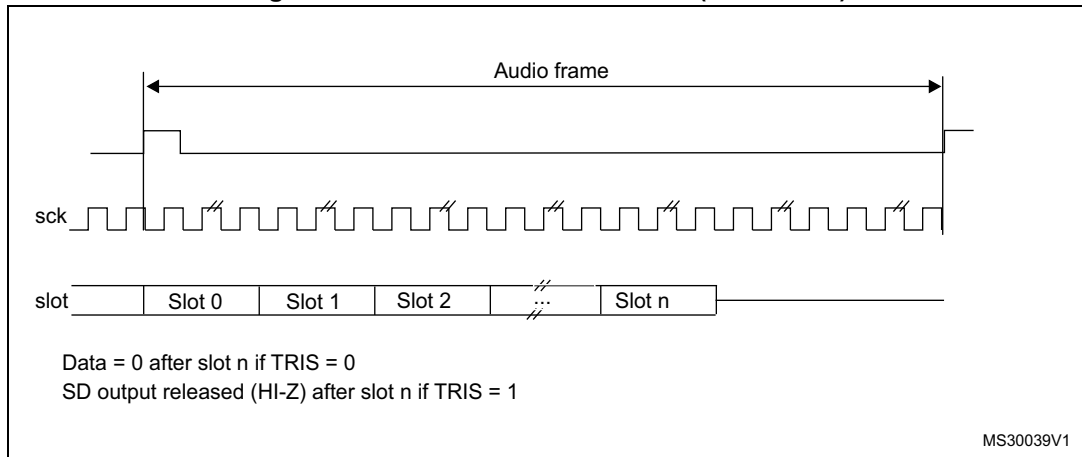


1. The frame length should be even.

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 will be forced to 0 until the end of frame in case of transmitter,
- if TRIS = 1, the line will be released to HI-Z during the transfer of these remaining bits. In reception mode, these bits are discarded.

Figure 512. 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 will be released and left free for other purposes.

### 47.3.7 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.

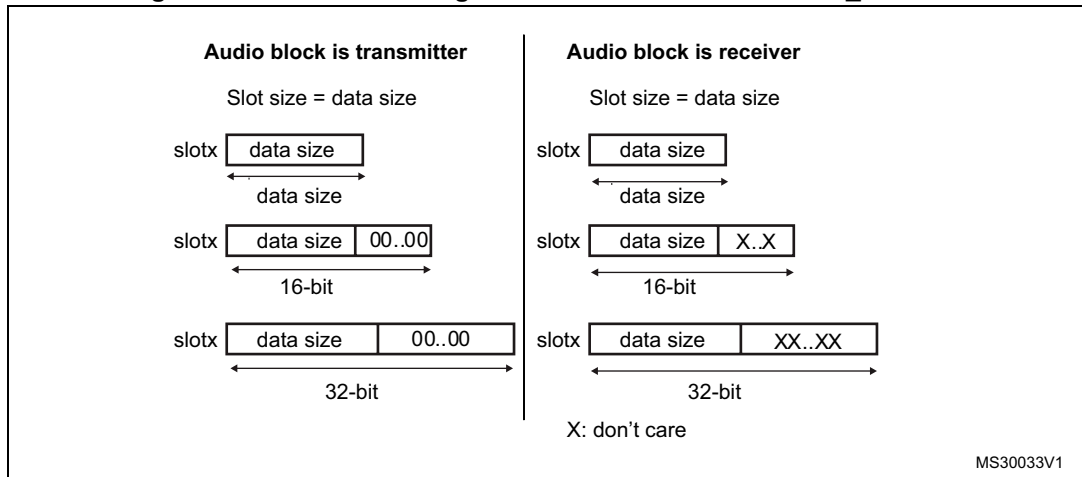
For AC'97 protocol or SPDIF (when bit  $PRTCFCFG[1:0] = 10$  or  $PRTCFCFG[1:0] = 01$ ), the number of slots is automatically set to target the protocol specification, and the value of  $NBSLOT[3:0]$  is ignored.

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 will be no FIFO access and so no request to read or write the FIFO linked to this inactive slot status.

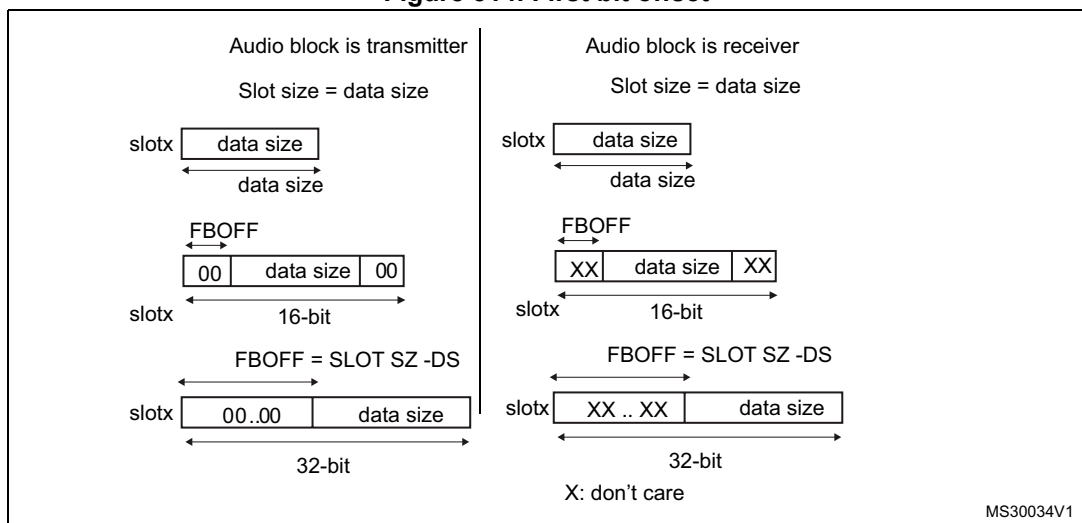
The slot size is also configurable as shown in [Figure 513](#). The size of the slots is selected by setting  $SLOTSZ[1:0]$  bits in the  $SAI\_xSLOTR$  register. The size is applied identically for each slot in an audio frame.

**Figure 513. 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 will be 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 514. 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 47.3.11: AC'97 link controller](#).

### 47.3.8 SAI clock generator

Each audio sub-block has its own clock generator that makes these two blocks completely independent. There is no difference in terms of functionality between these two clock generators.

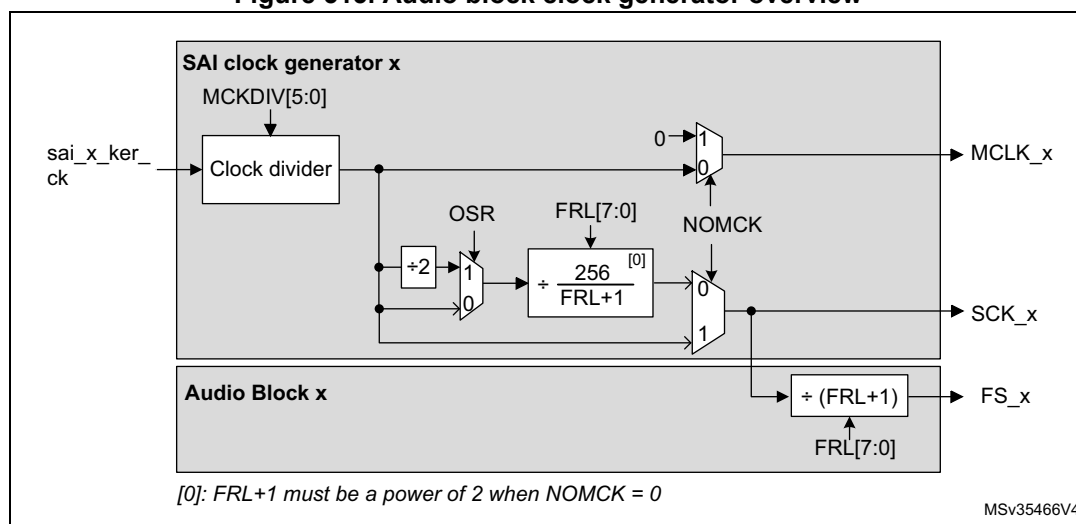
When the audio block is configured as master, the clock generator provides the bit clock (SCK\_x) and the master clock (MCLK\_x) for external decoders. The frame synchronization (FS\_x) is also derived from the signals provided by the clock generator. The clock source for the SAI clock generator (sai\_x\_ker\_ck) is delivered by the product clock controller (RCC).

When the audio block is defined as slave, the clock generator is OFF. The value of NOMCK, MCKDIV and OSR bits are ignored. In addition, MCLK\_x I/O pin is released and can be used as a general purpose I/O.

The bit clock strobing edge of (SCK\_x) can be configured through CKSTR bit in the SAI\_xCR1 register. This bit is functional in master and slave mode.

Figure 515 illustrates the architecture of the audio block clock generator.

Figure 515. Audio block clock generator overview



The NOMCK bit of the SAI\_xCR1 register is used to define whether the master clock is generated or not.

When the SAI is used in master mode, the clock generator configuration differs depending on whether a master clock (MCLK\_x) needs to be provided or not.

If NOMCK is set to 1, the master clock is not generated, and the user has more flexibility to select the frame length and frame synchronization frequency. In addition, MCLK\_x signal is driven Low if this pin is configured as the SAI pin in GPIO peripherals. MCKDIV can still be used to adjust the SCK\_x clock to the required frequency.

If NOMCK is set to 0, the master clock is generated, and can be used as reference clock for external decoders. In this case, the frequency ratio between the frame synchronization and the master clock is fixed to 512 or 256, and the frame length must be a power of 2. More details are given hereafter.

**Clock generator programming with MCLK (NOMCK = 0)**

In that case, MCLK\_x frequency will be:

$$F_{MCLK\_x} = 256 \times F_{FS\_x} \text{ if } OSR=0$$

$$F_{MCLK\_x} = 512 \times F_{FS\_x} \text{ if } OSR=1$$

When MCKDIV is different from 0, MCLK\_x frequency is given by the equation below:

$$F_{MCLK\_x} = \frac{F_{sai\_x\_ker\_ck}}{MCKDIV}$$

The frame synchronization frequency is given by:

$$F_{FS\_x} = \frac{F_{sia\_x\_ker\_ck}}{MCKDIV \times (OSR + 1) \times 256}$$

The frequency of the bit clock (SCK\_x) is given by the following expression:

$$F_{SCK\_x} = \frac{F_{MCLK\_x} \times (FRL + 1)}{(OSR + 1) \times 256}$$

*Note:* If  $NOMCK = 0$ ,  $(FRL+1)$  must be a power of two. In addition  $(FRL+1)$  must be between 8 and 256 (see [Section : FS signal role](#)).

*When MCKDIV division ratio is odd, the duty cycle of MCLK will not be 50%. The bit clock signal (SCK\_x) can also have a duty cycle different from 50% if MCKDIV is odd, and if OSR is equal to 0, and if  $(FRL+1) = 2^8$ .*

*It is recommended to configure MCKDIV to an even or big values (higher than 10).*

*Note that MCKDIV = 0 gives the same result as MCKDIV = 1.*

**Clock generator programming without MCLK (NOMCK = 1)**

When MCKDIV is different from 0, SCK\_x frequency is given in the equation below:

$$F_{SCK\_x} = \frac{F_{sai\_x\_ker\_ck}}{MCKDIV}$$

The frequency of the frame synchronization (FS\_x) is given by the following equation:

$$F_{FS\_x} = \frac{F_{sai\_x\_ker\_ck}}{(FRL + 1) \times MCKDIV}$$

*Note:* When  $NOMCK = 0$ ,  $(FRL+1)$  can take any values from 8 to 256.

*Note that MCKDIV = 0 gives the same result as MCKDIV = 1.*

### Clock generator programming examples

Table 282 shows some programming examples for 48, 96 and 192 kHz.

**Table 282. Clock generator programming examples**

Input sai_x_ker_ck clock frequency	MCLK	F <sub>MCLK</sub> /F <sub>FS</sub>	FRL <sup>(1)</sup>	OSR	NOMCK	MCKDIV[5:0]	Audio Sampling frequency (F <sub>FS</sub> )
98.304 MHz	Y	512	2 <sup>N</sup> -1	1	0	0 or 1	192 kHz
		512	2 <sup>N</sup> -1	1	0	2	96 kHz
		512	2 <sup>N</sup> -1	1	0	4	48 kHz
		256	2 <sup>N</sup> -1	0	0	2	192 kHz
		256	2 <sup>N</sup> -1	0	0	4	96 kHz
		256	2 <sup>N</sup> -1	0	0	8	48 kHz
	N	-	63	-	1	8	192 kHz
		-	63	-	1	16	96 kHz
		-	63	-	1	32	48 kHz

1. N is an integer value between 3 and 8.

### 47.3.9 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 interruption 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.

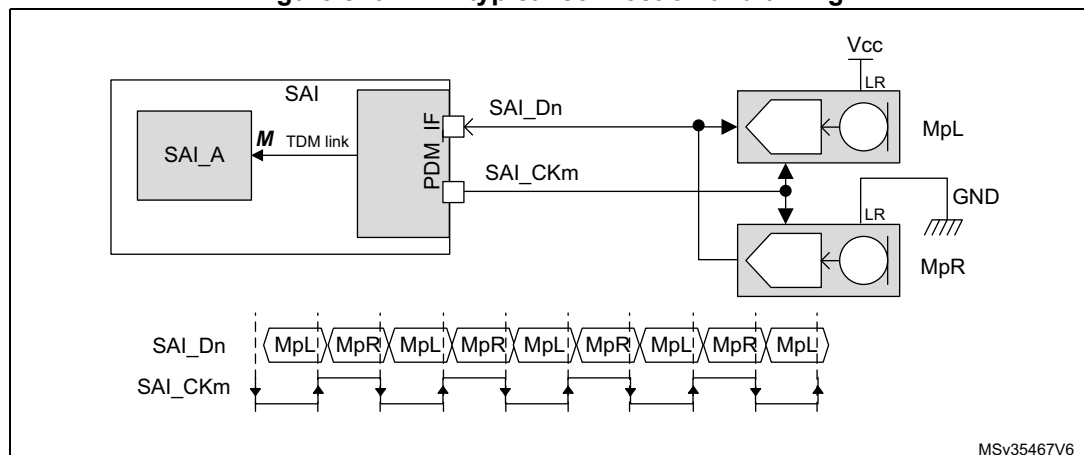
Data received will be right aligned in the SAI\_xDR.

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 will be lost automatically.

### 47.3.10 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. *Figure 516* 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 516. 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 sub-block configured in TDM MASTER mode. It cannot be used with SAI\_B sub-block. 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]).



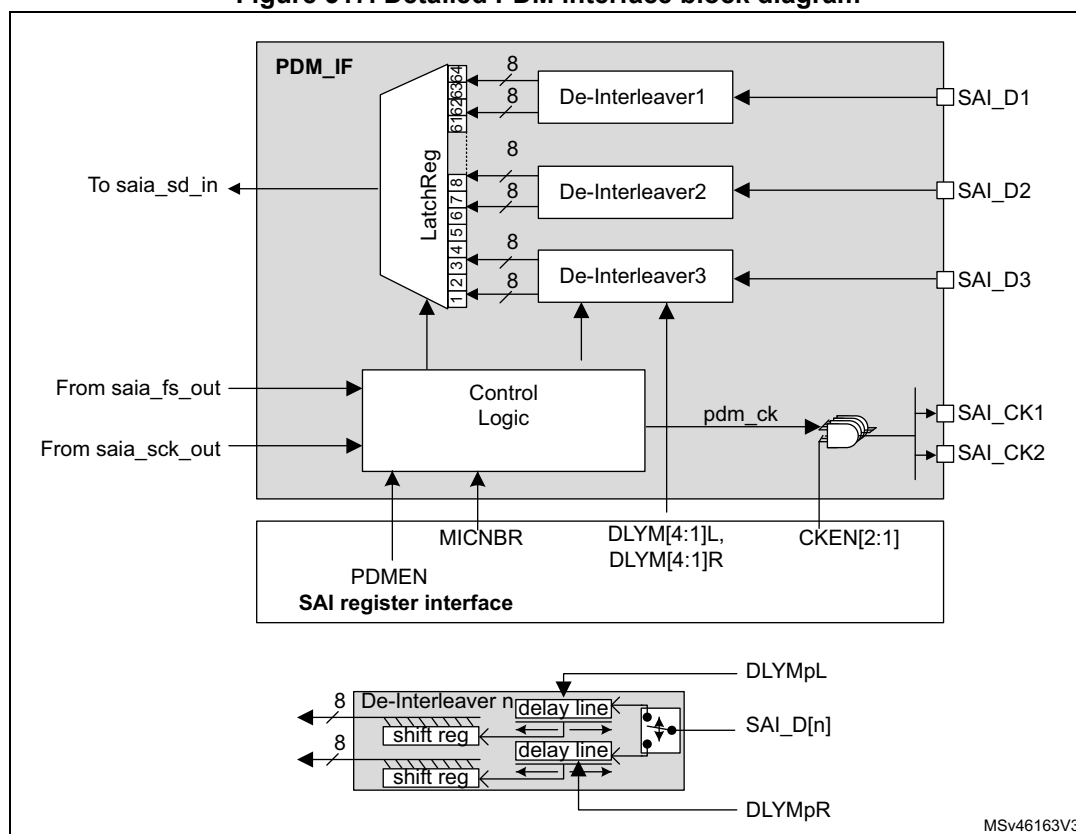
The data processing sequence into the PDM is the following:

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 517 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 517. Detailed PDM interface block diagram



1. *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:

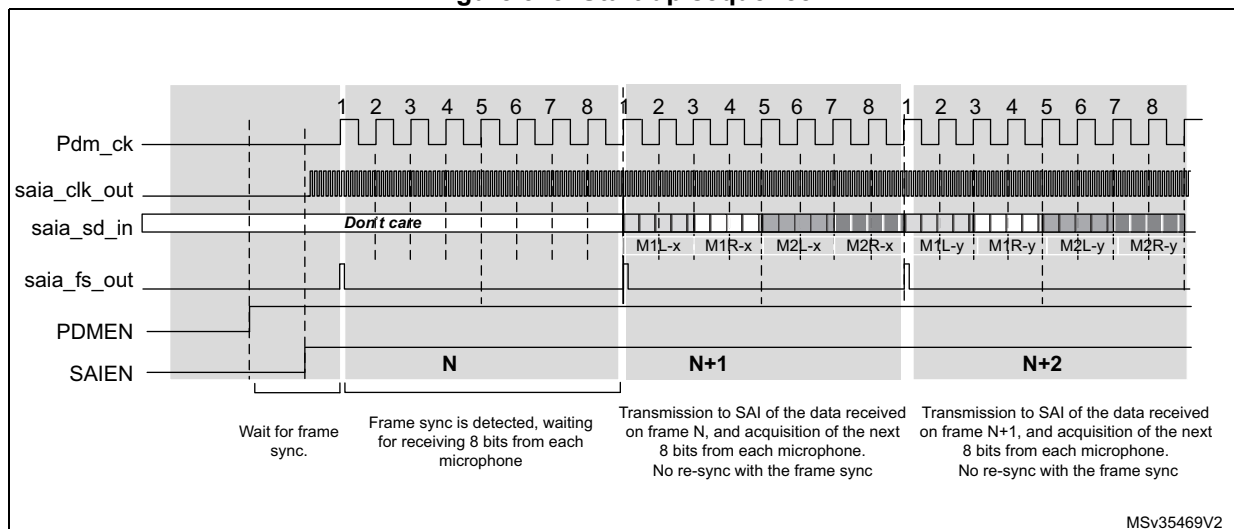
1. Configure SAI\_A in TDM MASTER mode (see [Table 283](#)).
2. Configure the PDM interface as follows:
  - a) Define the number of digital microphones via MICNBR.
  - b) Enable the bitstream clock needed in the application by setting the corresponding bits on CKEN to 1.
3. Enable the PDM interface, 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 shall be dropped.

### Start-up sequence

[Figure 518](#) 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\_CLK clock periods, a data byte coming from each microphone is available, and transferred to the SAI, via the TDM interface.

**Figure 518. 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 will contain 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 will 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.

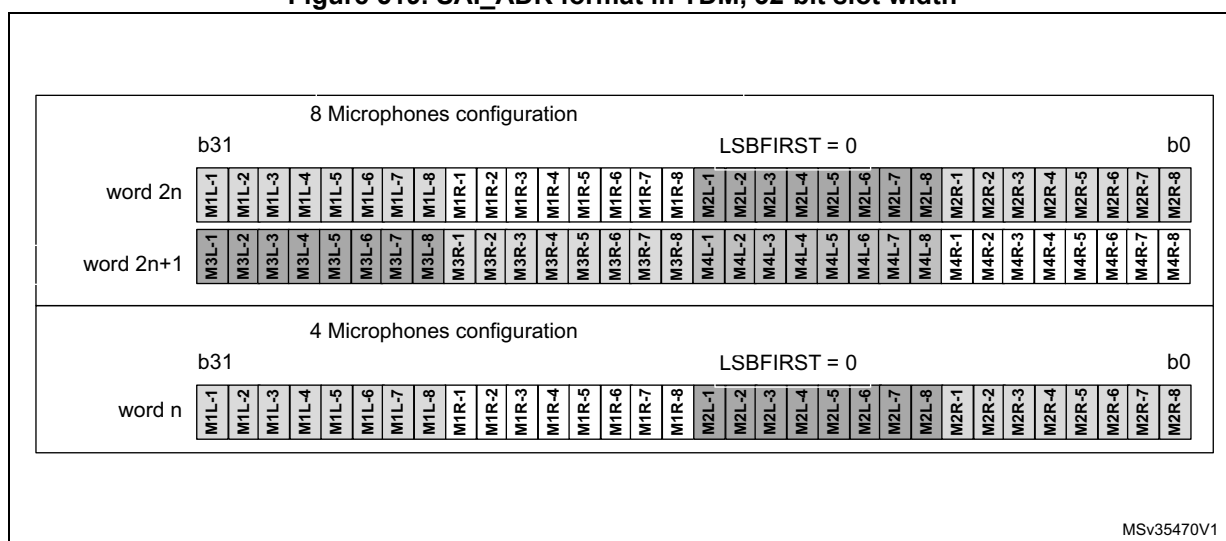
**SAI\_ADR data format example**

- **32-bit slot width** (DS = 0b111 and SLOTSZ = 0). Refer to [Figure 519](#).

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 519. SAI\_ADR format in TDM, 32-bit slot width**

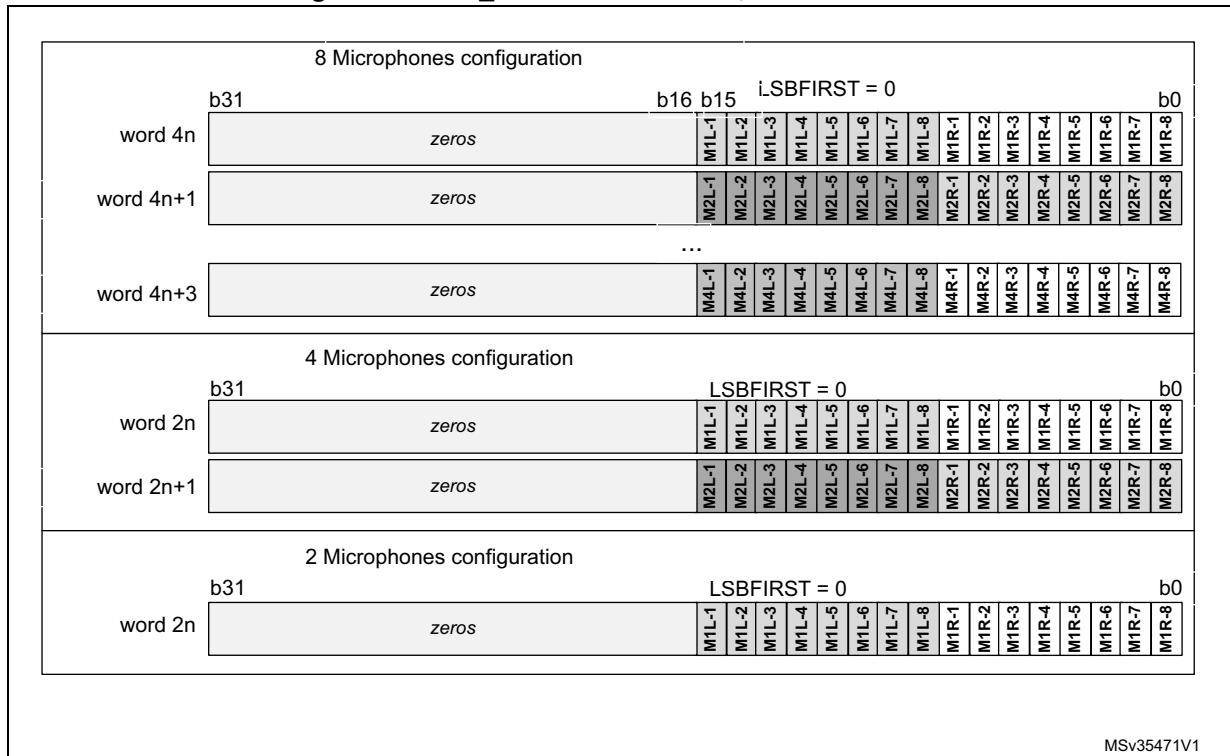


- **16-bit slot width** (DS = 0b100 and SLOTSZ = 0). Refer to [Figure 520](#).

For an 8 microphone configuration, four consecutive words read from the SAI\_ADR register contain a data byte from each microphone. Note that the 16-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 2 words of 16 bits are required to acquire a byte from 4 microphones and a single word for 2 microphones.

Figure 520. SAI\_ADR format in TDM, 16-bit slot width

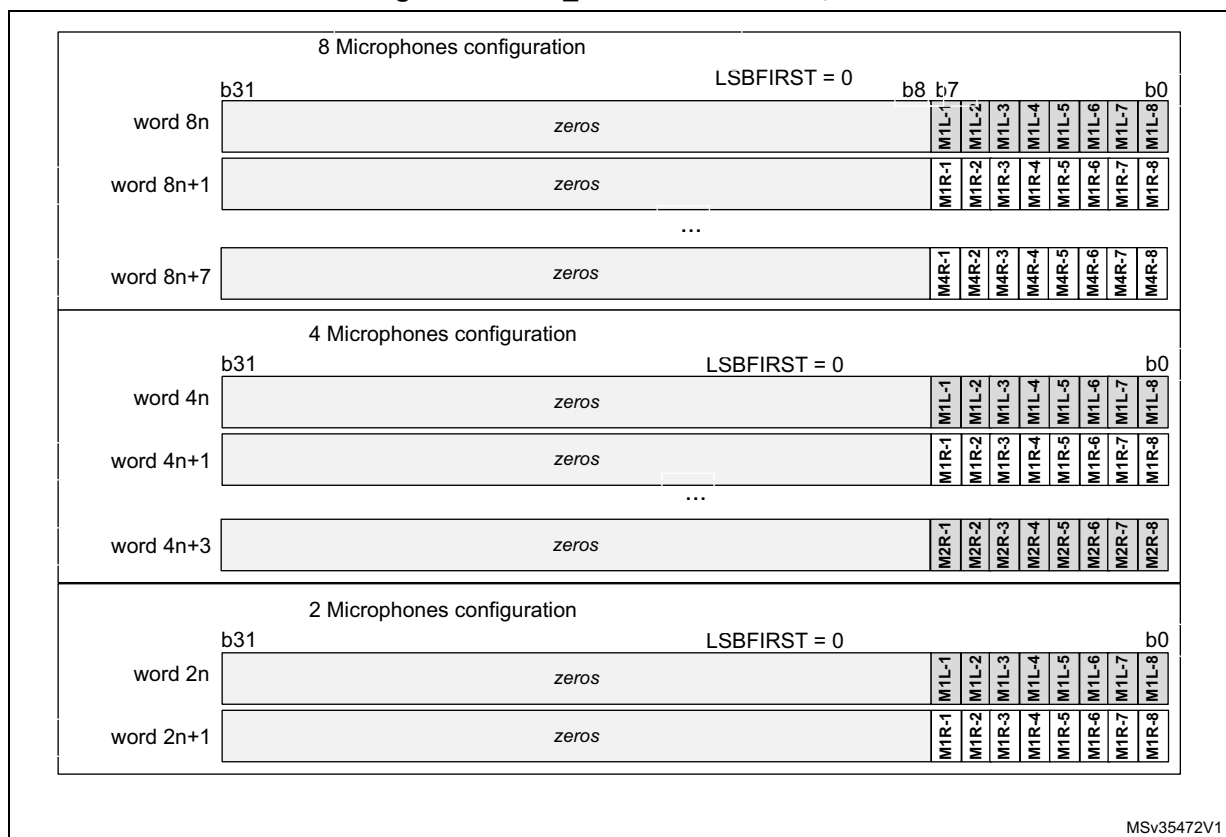


- Using a 8-bit slot width** (DS = 0b010 and SLOTSZ = 0). Refer to [Figure 521](#).

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 521. SAI\_ADR format in TDM, 8-bit slot width



MSv35472V1

**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 283](#) to ensure a good connection with the PDM interface.

Table 283. TDM settings

Bit Fields	Values	Comments
MODE	0b01	Mode must be MASTER receiver
PRTCFCG	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 284</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 284</a> .
FSALL	0	Pulse width is one bit clock cycle
FSDEF	0	FS signal is a start of frame

Table 283. 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 284</a> .
SLOTEN	X	To be adjusted according to NBSLOT
NOMCK	1	No need to generate a master clock MCLK
MCKDIV	X	Depends on the frequency provided to sai_a_ker_ck input. This parameter shall be adjusted to generate the proper bitstream clock frequency. See <a href="#">Table 284</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 283](#), and follow the below rules:

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 47.5.18](#))

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 284. Allowed TDM frame configuration<sup>(1)</sup>**

Microphone Sample rate	Number of microphones	Wanted SAI_CK[m] frequency <sup>(2)</sup>	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	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	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 283: TDM settings](#) for additional information on TDM configuration. The sai\_a\_ker\_ck clock frequency provided to the SAI should be a multiple of the SCK\_A frequency, and MCKDIV should be programmed accordingly.
2. 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 will become effective after two TDM frames.

### 47.3.11 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.

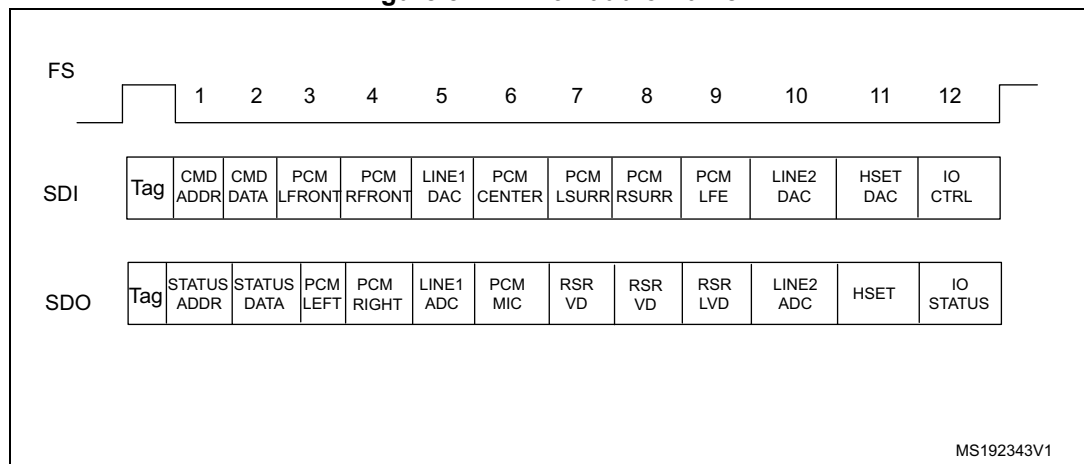
To select this protocol, set PRTCFCG[1:0] bits in the SAI\_xCR1 register to 10. When AC'97 mode is selected, only data sizes of 16 or 20 bits can be used, otherwise the SAI behavior is not guaranteed.

- 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\_xFRCCR 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 522 shows an AC'97 audio frame structure.

Figure 522. 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.

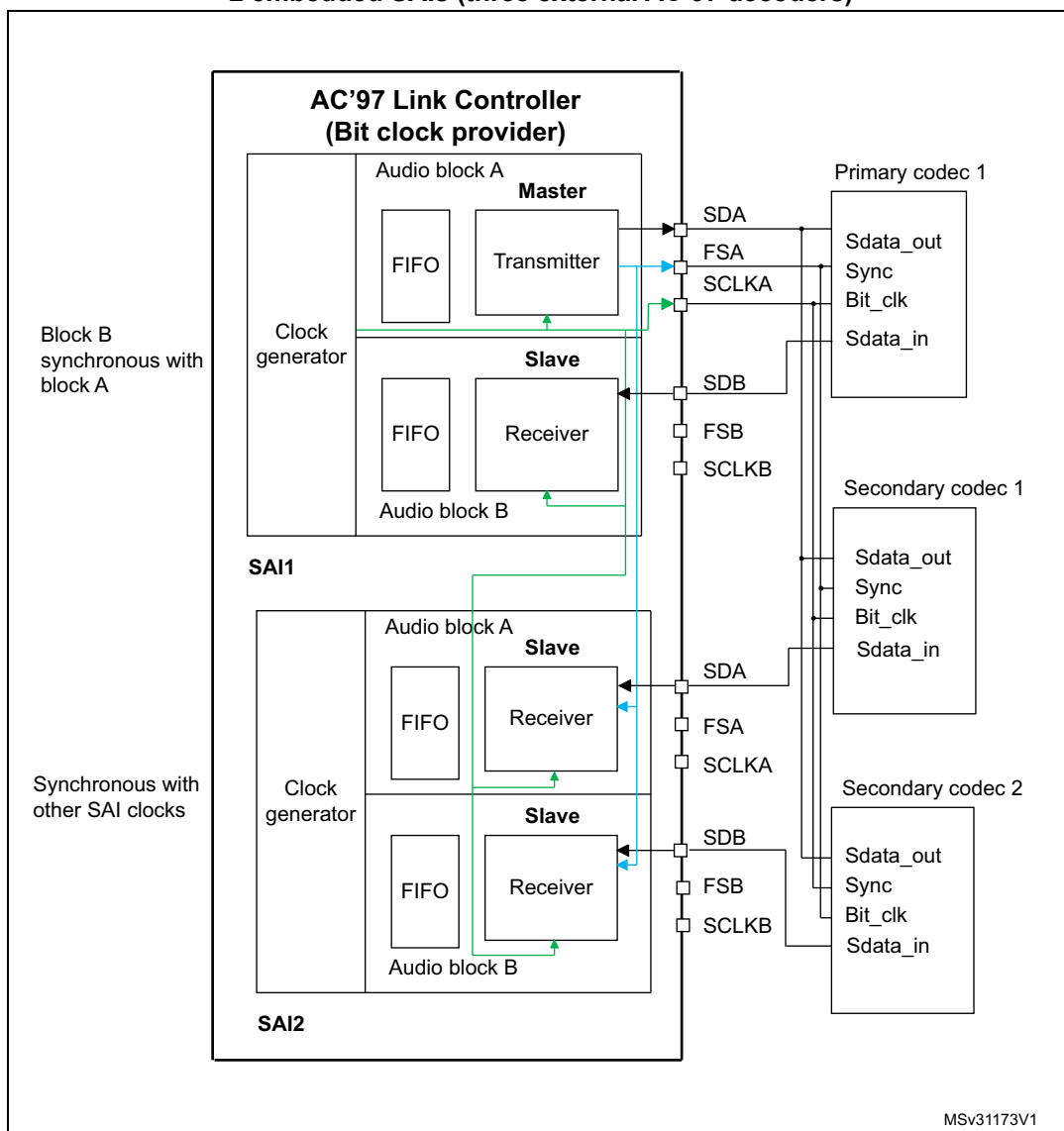
Using two SAIs (for devices featuring two embedded SAIs) allows controlling three external AC'97 decoders as illustrated in Figure 523.

In SAI1, the audio block A must be declared as asynchronous master transmitter whereas the audio block B is defined to be slave receiver and internally synchronous to the audio block A.

The SAI2 is configured for audio block A and B both synchronous with the external SAI1 in slave receiver mode.



**Figure 523. Example of typical AC'97 configuration on devices featuring at least 2 embedded SAIs (three external AC'97 decoders)**



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 will be 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 shall be set to 48 kHz. The formulas given in [Section 47.3.8: SAI clock generator](#) shall be used with  $FRL = 255$ , in order to generate the proper frame rate ( $F_{FS\_x}$ ).

### 47.3.12 SPDIF output

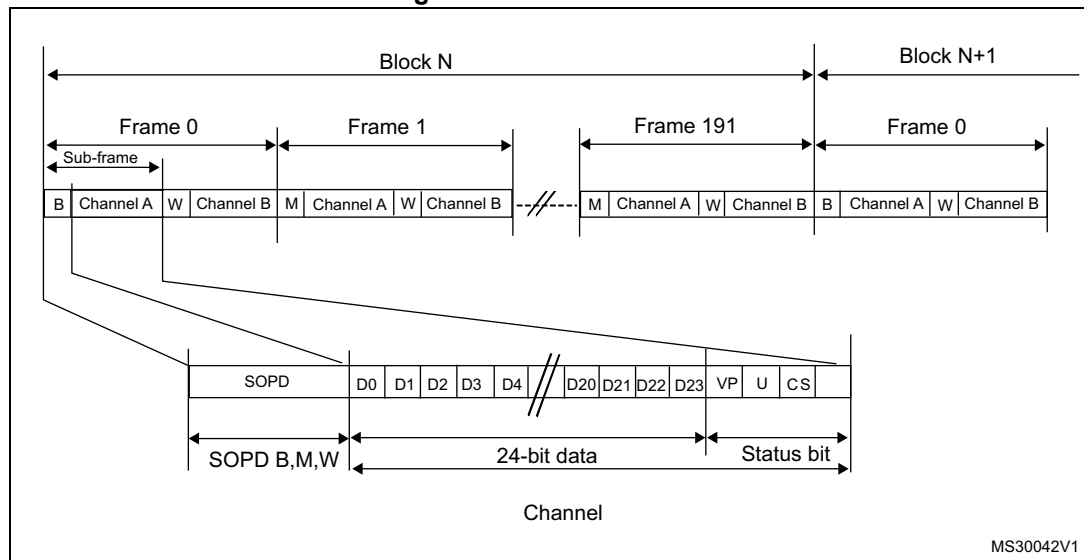
The SPDIF interface is available in transmitter mode only. It supports the audio IEC60958.

To select SPDIF mode, set PRTCFG[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 524](#).

Figure 524. 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 285](#)). The next 28 bits of channel information are composed of 24 bits data + 4 status bits.

**Table 285. 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:

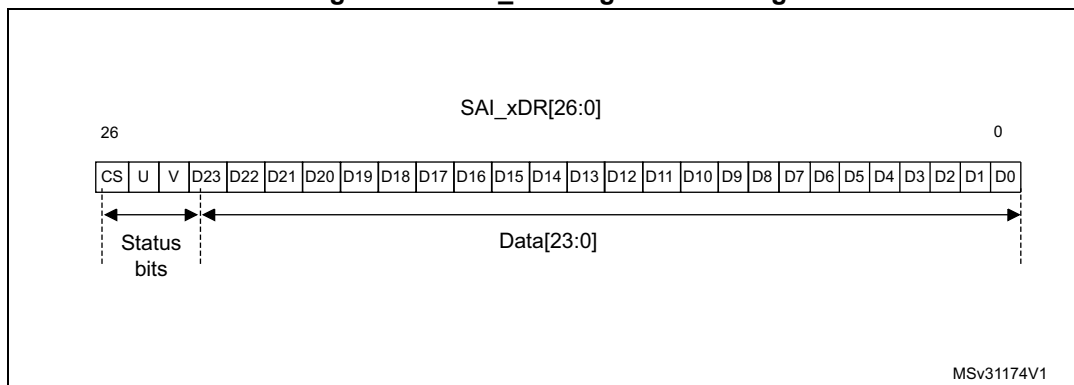
- SAI\_xDR[26:24] contain the Channel status, User and Validity bits.
- SAI\_xDR[23:0] contain the 24-bit data for the considered channel.

If the data size is 20 bits, then data shall be mapped on SAI\_xDR[23:4].

If the data size is 16 bits, then data shall be mapped on SAI\_xDR[23:8].

SAI\_xDR[23] always represents the MSB.

**Figure 525. 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 286](#).

**Table 286. 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 shall provide 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 287. Audio sampling frequency versus symbol rates**

Audio Sampling Frequencies (F <sub>S</sub> )	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<sub>S</sub>) and the bit clock rate (F<sub>SCK\_x</sub>) is given by the formula:

$$F_S = \frac{F_{SCK\_x}}{128}$$

And the bit clock rate is obtained as follow:

$$F_{SCK\_x} = \frac{F_{SAI\_CK\_x}}{MCKDIV}$$

### 47.3.13 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 sub-block is a transmitter or a receiver.

#### Audio sub-block 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 will still be 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 sub-block 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 sub-block 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 will be 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 will be stored in the FIFO. The data belonging to slot 1 will be 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 TDM 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 526. 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).

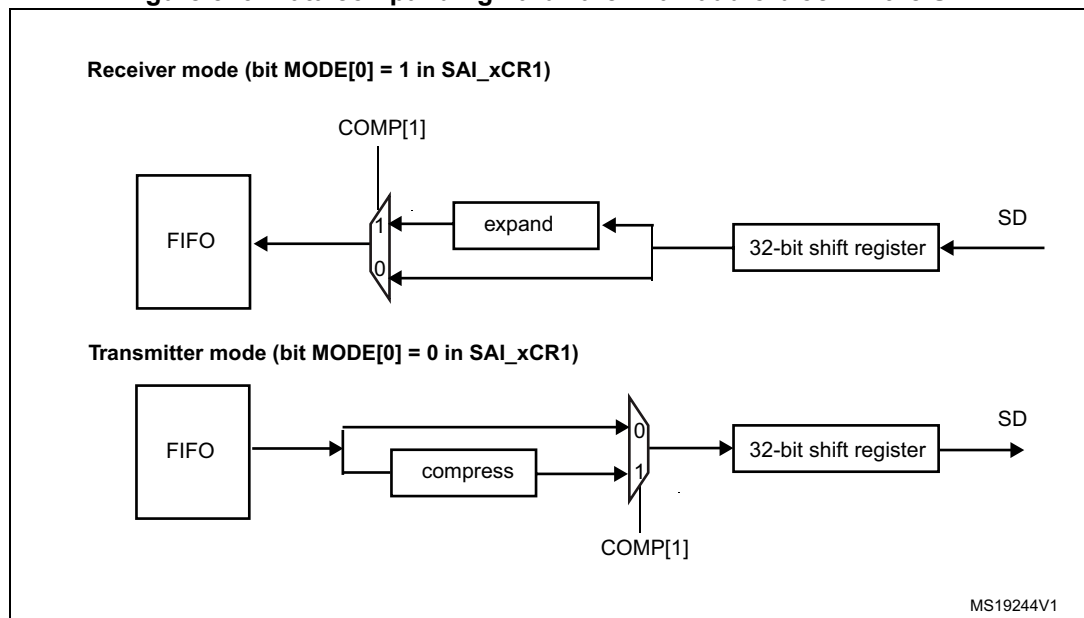
The European companding standard is A-Law and supports 13 bits of dynamic range (COMP[1:0] = 11 in the SAI\_xCR2 register).

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 will be 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 526. 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 will be applied.
- If the SAI audio block is declared as a receiver, the expansion algorithm will be 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).

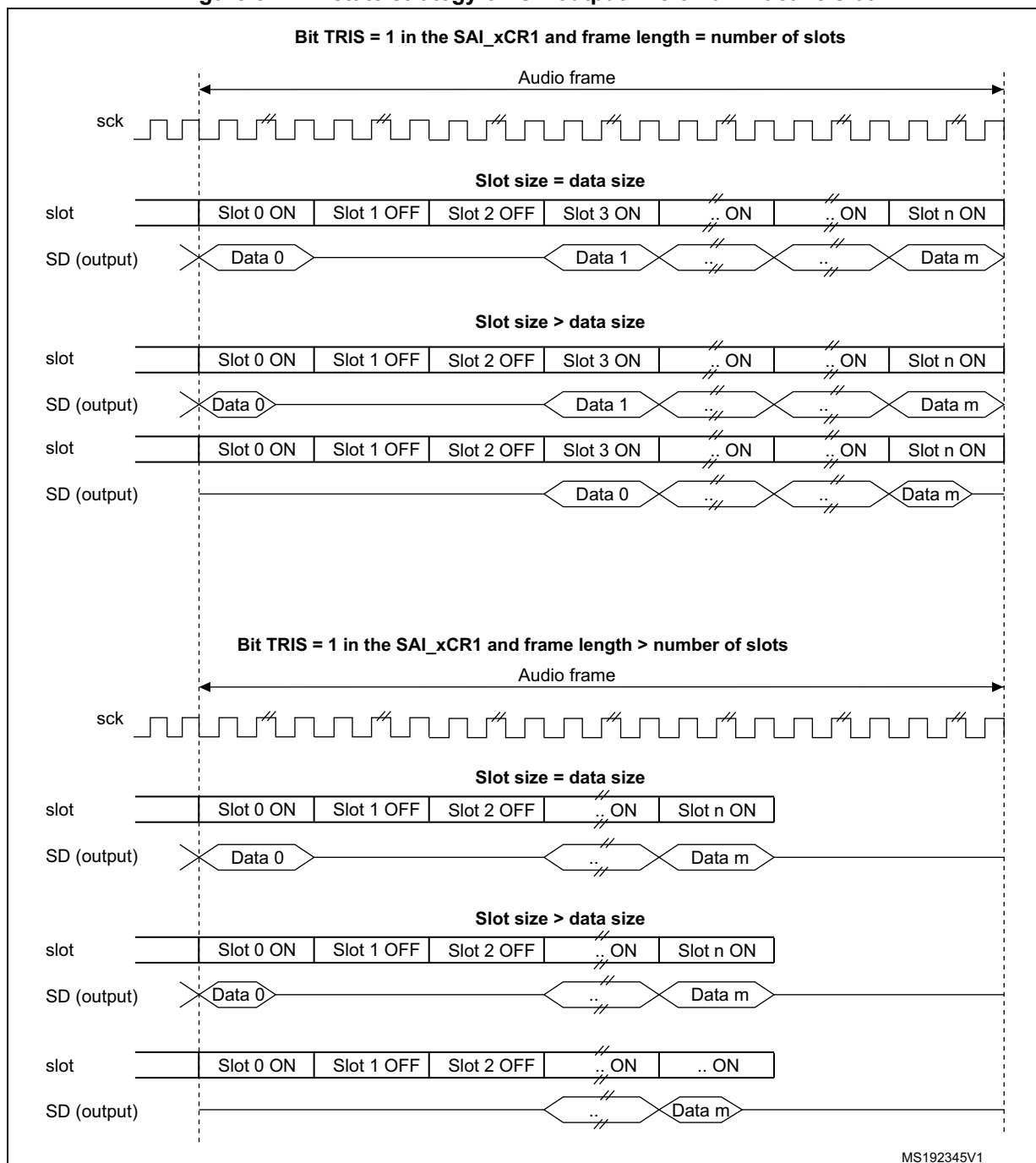
- Either the SAI forces 0 on the SD output line when an inactive slot is transmitted, or
- The line is released in HI-z state at the end of the last bit of data transferred, to release the line for other transmitters connected to this node.

It is important to note that the two transmitters cannot attempt to drive the same SD output pin simultaneously, which could 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 will then be 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 will be tri-stated when the padding to 0 is done to complete the audio frame.

*Figure 527* illustrates these behaviors.

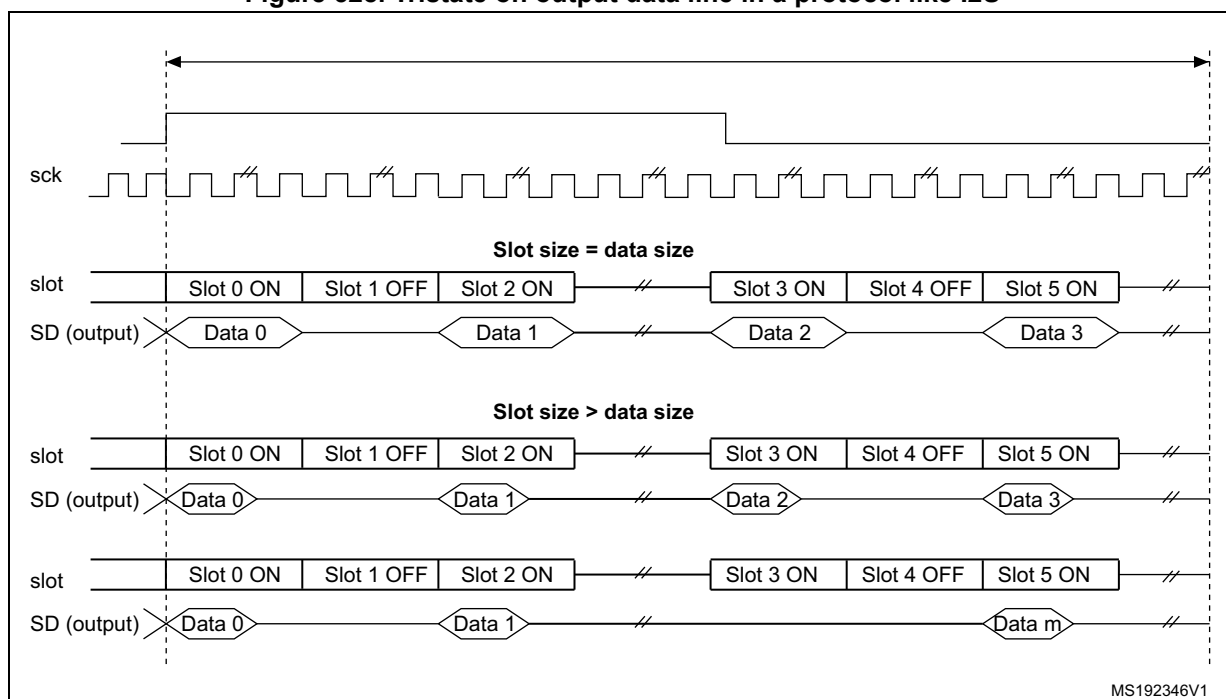
Figure 527. Tristate strategy on SD output line on an inactive slot



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 528](#) (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 528. 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 527](#) and [Figure 528](#) are replaced by a drive with a value of 0.

### 47.3.14 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.

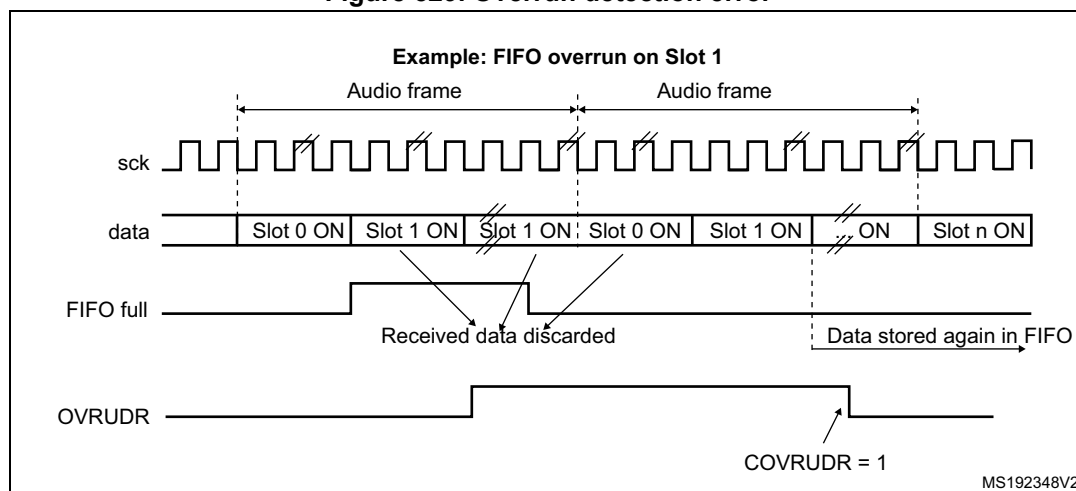
The overrun or underrun errors share the same bit since an audio block can be either receiver or transmitter and each audio block in a given SAI has its own SAI\_xSR register.

#### 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 will be 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 will store 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 529](#)).

The OVRUDR flag is cleared when COVRUDR bit is set in the SAI\_xCLRFR register.

**Figure 529. Overrun detection error**



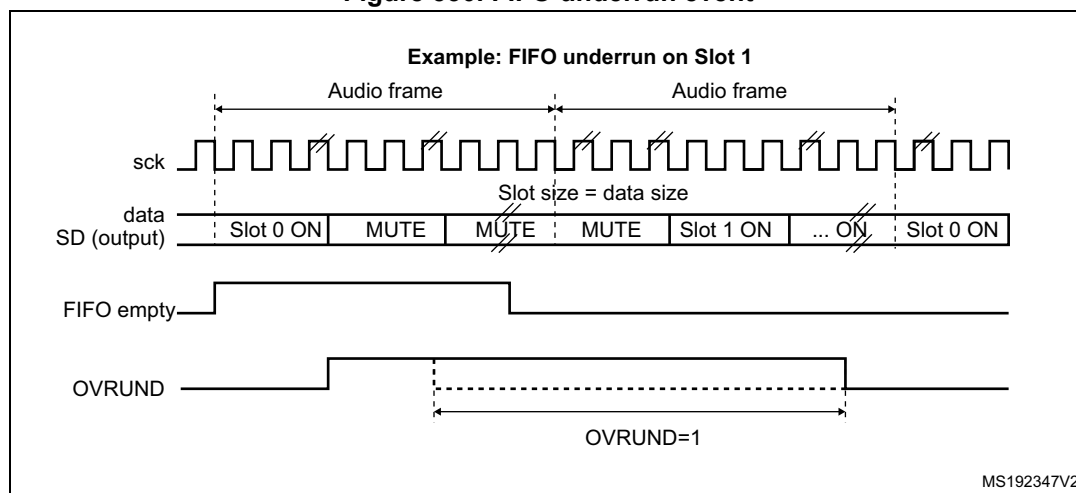
**Underrun**

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 530](#)). 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 sub-block is configured as master or slave.

**Figure 530. 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 will wait 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 will be 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 (PRTCFCFG[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 will be 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 will be 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 NOMCK = 0)

When the audio block operates as a master (MODE[1] = 0) and NOMCK 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, NOMCK, 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.

## 47.3.15 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.

## 47.3.16 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 sub-block supporting basic DMA request/acknowledge protocol.

To configure the audio sub-block 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 47.3.9: Internal FIFOs](#)). DMA transfer direction is linked to the SAI audio sub-block 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 will be 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.

## 47.4 SAI interrupts

The SAI supports 7 interrupt sources as shown in [Table 288](#).

**Table 288. SAI interrupt sources**

Interrupt source	Interrupt group	Audio block mode	Interrupt enable	Interrupt clear
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 47.3.9: Internal FIFOs</a>
OVRUDR	ERROR	Master or slave Receiver or transmitter	OVRUDRIE in SAI_xIM register	COVRUDR = 1 in SAI_xCLRFR register
AFSDDET	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 NOMCK = 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.

## 47.5 SAI registers

### 47.5.1 Global configuration register (SAI\_GCR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCOUT[1:0]		Res.	Res.	SYNCIN[1:0]	
										rw	rw			rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **SYNCOUT[1:0]**: Synchronization outputs

These bits are set and cleared by software.

00: No synchronization output signals. SYNCOUT[1:0] should be configured as “No synchronization output signals” when audio block is configured as SPDIF

01: Block A used for further synchronization for others SAI

10: Block B used for further synchronization for others SAI

11: Reserved. These bits must be set when both audio block (A and B) are disabled.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **SYNCIN[1:0]**: Synchronization inputs

These bits are set and cleared by software.

Refer to [Table 281: External synchronization selection](#) for information on how to program this field.

These bits must be set when both audio blocks (A and B) are disabled.

They are meaningful if one of the two audio blocks is defined to operate in synchronous mode with an external SAI (SYNCEN[1:0] = 10 in SAI\_ACR1 or in SAI\_BCR1 registers).

### 47.5.2 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.	Res.	OSR	MCKDIV[5:0]						NOMCK	Res.	DMAEN	SAIEN
					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.	PRTCFIG[1:0]		MODE[1:0]	
		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 **OSR**: Oversampling ratio for master clock

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 47.3.8: 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 **NOMCK**: No divider

This bit is set and cleared by software.

0: Master clock generator is enabled

1: Master clock generator is disabled. The clock divider controlled by MCKDIV can still be used to generate the bit clock.

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 will not be taken into account.

This bit allows controlling 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 **OUTDRIV**: 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 sub-block is disabled.

00: audio sub-block in asynchronous mode.

01: audio sub-block is synchronous with the other internal audio sub-block. In this case, the audio sub-block must be configured in slave mode

10: audio sub-block is synchronous with an external SAI embedded peripheral. In this case the audio sub-block should be configured in Slave mode.

11: Reserved

*Note: The audio sub-block 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 PRTCFG[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 **PRTCFG[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 allows 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).*

### 47.5.3 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.	Res.	OSR	MCKDIV[5:0]					NOMCK	Res.	DMAEN	SAIEN	
					r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w		r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]	CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]		
		r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w		r/w	r/w	r/w	r/w	

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **OSR**: Oversampling ratio for master clock

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 47.3.8: 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 **NOMCK**: No divider

This bit is set and cleared by software.

0: Master clock generator is enabled

1: Master clock generator is disabled. The clock divider controlled by MCKDIV can still be used to generate the bit clock.

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 will not be taken into account.

This bit allows controlling 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 **OUTDRIV**: 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 sub-block is disabled.

00: audio sub-block in asynchronous mode.

01: audio sub-block is synchronous with the other internal audio sub-block. In this case, the audio sub-block must be configured in slave mode

10: audio sub-block is synchronous with an external SAI embedded peripheral. In this case the audio sub-block should be configured in Slave mode.

11: Reserved

*Note: The audio sub-block 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 PRTCFG[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 PRTCFG[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 allows 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). In Master transmitter mode, the audio block starts generating the FS and the clocks immediately.*

### 47.5.4 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	MUTE CNT[5:0]					MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	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 will be 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 TDM 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 **MUTE CNT[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* will be set and an interrupt will be 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/interruption must be disabled

**Bits 2:0 FTH[2:0]:** FIFO threshold.

This bit is set and cleared by software.

000: FIFO empty

001: ¼ FIFO

010: ½ FIFO

011: ¾ FIFO

100: FIFO full

101: Reserved

110: Reserved

111: Reserved

### 47.5.5 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	MUTE CNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	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 will be 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 TDM 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 **MUTE CNT[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* will be set and an interrupt will be 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/interruption must be disabled

**Bits 2:0 FTH[2:0]:** FIFO threshold.

This bit is set and cleared by software.

000: FIFO empty

001: ¼ FIFO

010: ½ FIFO

011: ¾ FIFO

100: FIFO full

101: Reserved

110: Reserved

111: Reserved

### 47.5.6 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 will be 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 will 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 (NOMCK = 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.

### 47.5.7 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 will be 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 will 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 (NOMCK = 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.

### 47.5.8 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 will be undetermined.

Refer to [Section : 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.

### 47.5.9 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

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 will be undetermined.  
 Refer to [Section : 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.

### 47.5.10 Interrupt mask register 2 (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.	Res.	Res.	Res.	Res.	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.	LFSDET IE	AFSDET IE	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 will be 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 will be 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 interruption 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 interruption 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 NOMCK = 0.

It generates an interrupt if the WCKCFG flag in the SAI\_xSR register is set.

*Note: This bit is used only in TDM 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.

### 47.5.11 Interrupt mask register 2 (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.	Res.	Res.	Res.	Res.	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.	LFSDET IE	AFSDET IE	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 will be 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 will be 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 interruption 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 interruption 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 NOMCK = 0.

It generates an interrupt if the WCKCFG flag in the SAI\_xSR register is set.

*Note: This bit is used only in TDM 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.

### 47.5.12 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.	Res.	Res.	Res.	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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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).

If the SAI block is configured as transmitter:

000: FIFO empty

001: FIFO  $\leq \frac{1}{4}$  but not empty

010:  $\frac{1}{4} < \text{FIFO} \leq \frac{1}{2}$

011:  $\frac{1}{2} < \text{FIFO} \leq \frac{3}{4}$

100:  $\frac{3}{4} < \text{FIFO}$  but not full

101: FIFO full

If SAI block is configured as receiver:

000: FIFO empty

001: FIFO  $< \frac{1}{4}$  but not empty

010:  $\frac{1}{4} \leq \text{FIFO} < \frac{1}{2}$

011:  $\frac{1}{2} \leq \text{FIFO} < \frac{3}{4}$

100:  $\frac{3}{4} \leq \text{FIFO}$  but not full

101: FIFO full

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 47.3.6: 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 NOMCK = 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.

### 47.5.13 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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).

If the SAI block is configured as transmitter:

000: FIFO empty

001: FIFO  $\leq \frac{1}{4}$  but not empty

010:  $\frac{1}{4} < \text{FIFO} \leq \frac{1}{2}$

011:  $\frac{1}{2} < \text{FIFO} \leq \frac{3}{4}$

100:  $\frac{3}{4} < \text{FIFO}$  but not full

101: FIFO full

If SAI block is configured as receiver:

000: FIFO empty

001: FIFO  $< \frac{1}{4}$  but not empty

010:  $\frac{1}{4} \leq \text{FIFO} < \frac{1}{2}$

011:  $\frac{1}{2} \leq \text{FIFO} < \frac{3}{4}$

100:  $\frac{3}{4} \leq \text{FIFO}$  but not full

101: FIFO full

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 47.3.6: 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 NOMCK = 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.

### 47.5.14 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.	Res.	Res.	Res.	Res.	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.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTE DET	COVRUD R
										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 AFSDDET 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 **WCKCFG**: 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 NOMCK = 0 in the SAI\_xCR1 register.

Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.

This bit is write only.

Programming this bit to 1 clears the MUTEDET 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.

### 47.5.15 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.	Res.	Res.	Res.	Res.	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.	CLFSDET	CAFSDDET	CCNRDY	Res.	CWCKCFG	CMUTE DET	COVRUD R
									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 **CAFSDET**: 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 **WCKCFG**: 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 NOMCK = 0 in the SAI\_xCR1 register.

Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.

This bit is write only.

Programming this bit to 1 clears the MUTEDET 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.

### 47.5.16 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.

### 47.5.17 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.

### 47.5.18 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.	Res.	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.*

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.*

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.*

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 allows 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

### 47.5.19 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		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.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		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<sub>SAI\_CK</sub> period  
 010: Delay of 2 T<sub>SAI\_CK</sub> periods  
 ...  
 111: Delay of 7 T<sub>SAI\_CK</sub> periods

This field can be changed on-the-fly.

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<sub>SAI\_CK</sub> period  
 010: Delay of 2 T<sub>SAI\_CK</sub> periods  
 ...  
 111: Delay of 7 of T<sub>SAI\_CK</sub> periods

This field can be changed on-the-fly.

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.

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.

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.

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.

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.

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.0

### 47.5.20 SAI register map

The following table summarizes the SAI registers.

**Table 289. 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	SAI_GCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x0004 or 0x0024	SAI_xCR1	Res.	Res.	Res.	Res.	Res.	OSR	MCKDIV[3:0]					NOMCK	Res.	DMAEN	SAIEN	Res.	Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	Res.	DS[2:0]	Res.	PRTCFG[1:0]		MODE[1:0]				
	Reset value						0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	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.	COMP[1:0]	CPL	MUTECCN[5:0]					MUTE VAL	MUTE	TRIS	FFLUS	FTH					
	Reset value																		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.	FSOFF	FSPOL	FSDEF	Res.	FSALL[6:0]					FRL[7:0]											
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	
0x0010 or 0x0030	SAI_xSLOTR	SLOTEN[15:0]															Res.	Res.	Res.	Res.	NBSLOT[3:0]			SLOTSZ[1:0]		Res.	FBOFF[4:0]								
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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.	LFSDET	AFSDETIE	CNRDYIE	FREQIE	WCKCFG	MUTEDET	OVRUDRIE		
	Reset value																										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.	FLV[2:0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR		
	Reset value															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.	LFSDET	CAFSDET	CNRDY	Res.	WCKCFG	MUTEDET	OVRUDR			
	Reset value																									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	0	0	0



Table 289. 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	
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.	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	Res.	PDMEN
	Reset value																							0	0			0	0					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	

Refer to [Section 2.2 on page 81](#) for the register boundary addresses.

## 48 Secure digital input/output MultiMediaCard interface (SDMMC)

### 48.1 SDMMC main features

The SD/SDIO, embedded MultiMediaCard (e•MMC) host interface (SDMMC) provides an interface between the AHB bus and SD memory cards, SDIO cards and e•MMC devices.

The MultiMediaCard system specifications are available through the MultiMediaCard Association website at [www.mmca.org](http://www.mmca.org), published by the MMCA technical committee.

SD memory card and SD I/O card system specifications are available through the SD card Association website at [www.sdcard.org](http://www.sdcard.org).

The SDMMC features include the following:

- Compliance with *Embedded MultiMediaCard System Specification Version 4.51*. Card support for three different databus modes: 1-bit (default), 4-bit and 8-bit.
- Full compatibility with previous versions of MultiMediaCards (backward compatibility).
- Full compliance with *SD memory card specifications version 4.1*. (SPI mode and UHS-II mode not supported).
- Full compliance with *SDIO card specification version 4.0*. Card support for two different databus modes: 1-bit (default) and 4-bit. (SPI mode and UHS-II mode not supported).
- Data transfer up to 104 Mbyte/s for the 8-bit mode. (depending maximum allowed I/O speed).
- Data and command output enable signals to control external bidirectional drivers.

The MultiMediaCard/SD bus connects cards to the host.

The current version of the SDMMC supports only one SD/SDIO/e•MMC card at any one time and a stack of e•MMC.

### 48.2 SDMMC bus topology

Communication over the bus is based on command/response and data transfers.

The basic transaction on the SD/SDIO/e•MMC bus is the command/response transaction. These types of bus transaction transfer their information directly within the command or response structure. In addition, some operations have a data token.

Data transfers are done in the following ways:

- Block mode: data block(s) with block size  $2^N$  bytes with N in the range 0-14.
- SDIO multibyte mode: single data block with block size range 1-512 bytes
- e•MMC Stream mode: continuous data stream

Data transfers to/from e•MMC cards are done in data blocks or streams.

Figure 531. SDMMC “no response” and “no data” operations

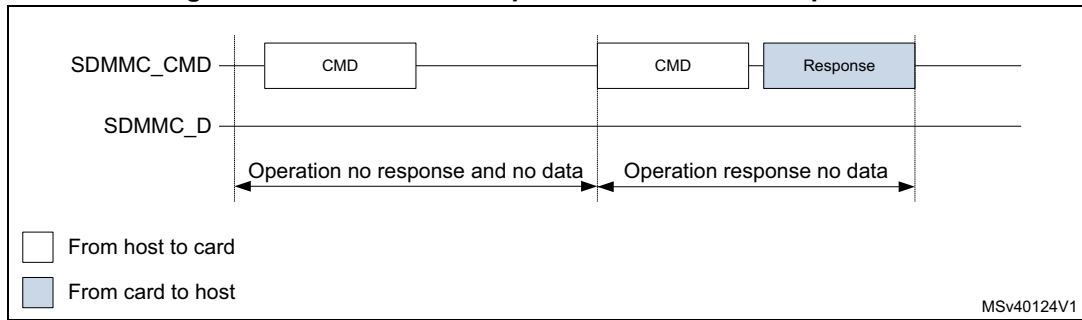
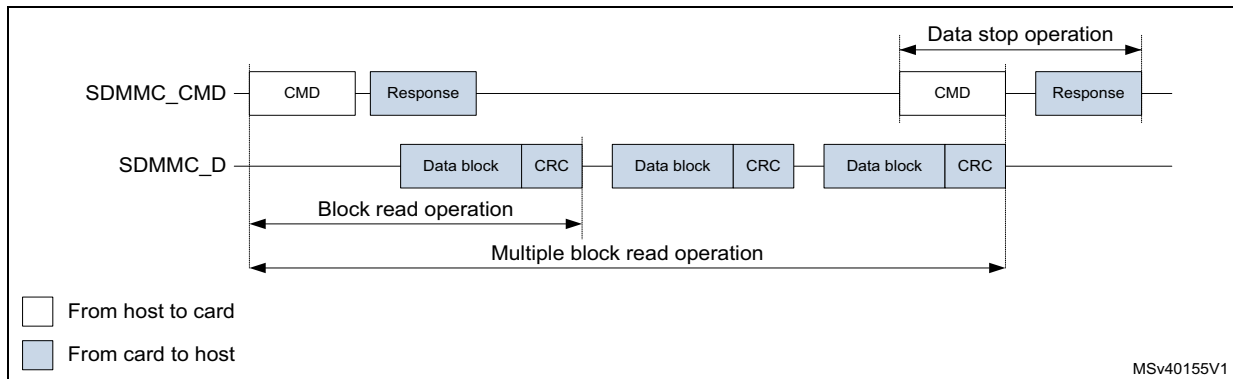
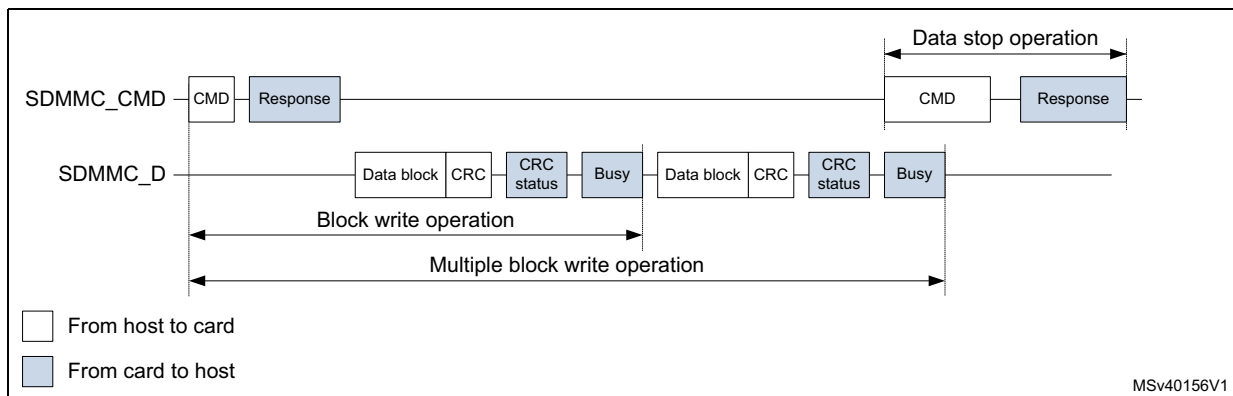


Figure 532. SDMMC (multiple) block read operation



Note: The Stop Transmission command is not required at the end of a eMMC multiple block read with predefined block count.

Figure 533. SDMMC (multiple) block write operation



Note: The Stop Transmission command is not required at the end of an eMMC multiple block write with predefined block count.

Note: The SDMMC will not send any data as long as the Busy signal is asserted (SDMMC\_D0 pulled low).

Figure 534. SDMMC (sequential) stream read operation

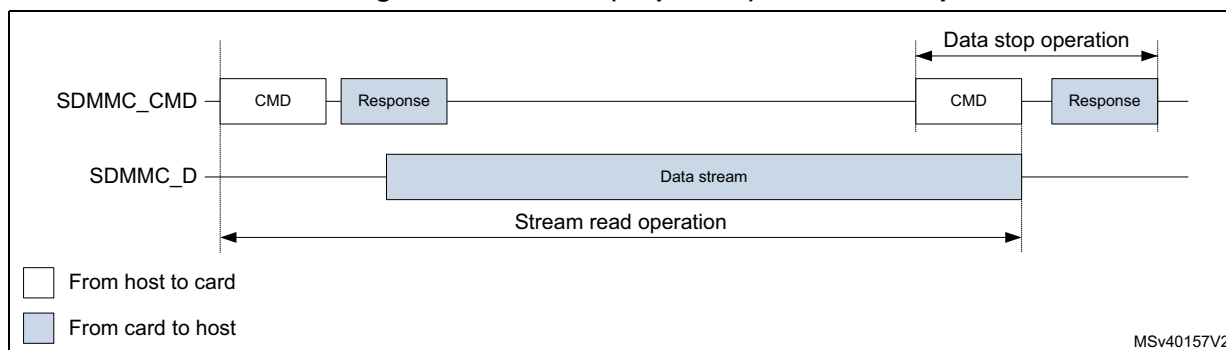
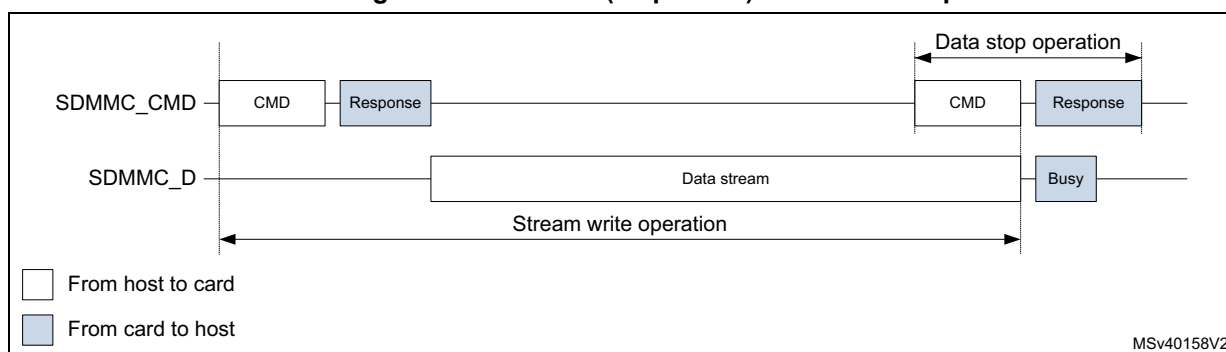


Figure 535. SDMMC (sequential) stream write operation



Stream data transfer operates only in a 1-bit wide bit bus configuration on SDMMC\_D0 in single data rate modes (DS, HS, and SDR).

### 48.3 SDMMC operation modes

Table 290. SDMMC operation modes SD & SDIO

SDIO Bus Speed modes <sup>(1)(2)</sup>	Max Bus Speed <sup>(3)</sup> [MByte/s]	Max Clock frequency [MHz] <sup>(4)</sup>	Signal Voltage [V]
DS (Default Speed)	12.5	25	3.3
HS (High Speed)	25	50	3.3
SDR12	12.5	25	1.8
SDR25	25	50	1.8
DDR50	50	50	1.8
SDR50	50	100	1.8

1. SDR single data rate signaling.
2. DDR double data rate signaling. (data is sampled on both SDMMC\_CK clock edges).
3. SDIO bus speed with 4bit bus width.
4. Maximum frequency depending on maximum allowed IO speed.

Table 291. SDMMC operation modes eMMC

eMMC bus speed modes <sup>(1)(2)</sup>	Max bus speed <sup>(3)</sup> [MByte/s]	Max clock frequency [MHz] <sup>(4)</sup>	Signal voltage [V]
Legacy compatible	26	26	3/1.8/1.2V
High speed SDR	52	52	3/1.8/1.2V
High speed DDR	104	52	3/1.8/1.2V

1. SDR single data rate signaling.
2. DDR double data rate signaling. (data is sampled on both SDMMC\_CK clock edges).
3. eMMC bus speed with 8bit bus width.
4. Maximum frequency depending on maximum allowed IO speed.

## 48.4 SDMMC functional description

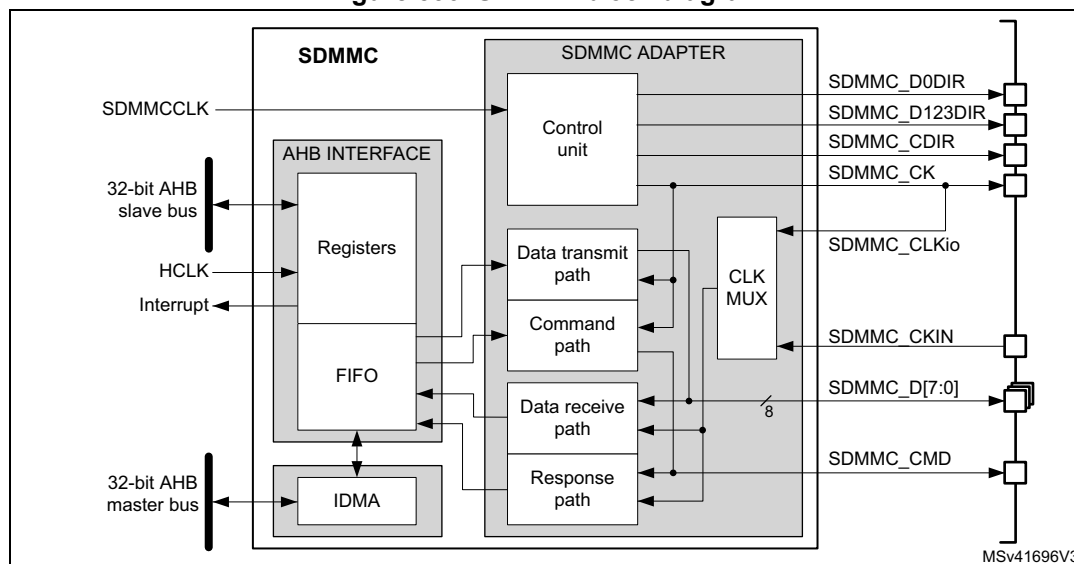
The SDMMC consists of three parts:

- The AHB slave interface accesses the SDMMC adapter registers, and generates interrupt signals and IDMA control signals.
- The SDMMC adapter block provides all functions specific to the eMMC/SD/SD I/O card such as the clock generation unit, command and data transfer.
- The internal DMA (IDMA) block with its AHB master interface.

### 48.4.1 SDMMC diagram

Figure 536 shows the SDMMC block diagram.

Figure 536. SDMMC block diagram



### 48.4.2 SDMMC pins and internal signals

Table 292 lists the SDMMC internal input/output signals, Table 293 the SDMMC pins (alternate functions).

Table 292. SDMMC internal input/output signals

Signal name	Signal type	Description
sdmmc_ker_ck	Digital input	SDMMC kernel clock
sdmmc_hclk	Digital input	AHB clock
sdmmc_it	Digital output	SDMMC global interrupt
sdmmc_dataend_trg	Digital output	SDMMC data end trigger for MDMA
sdmmc_io_in_ck	Digital input	SD/SDIO/eMMC card feedback clock. This signal is internally connected to the SDMMC_CK pin (for DS and HS modes).

Table 293. SDMMC pins

Signal name	Signal type	Description
SDMMC_CK	Digital output	Clock to SD/SDIO/eMMC card
SDMMC_CKIN	Digital input	Clock feedback from an external driver for SD/SDIO/eMMC card. (for SDR12, SDR25, SDR50, DDR50)
SDMMC_CMD	Digital input/output	SD/SDIO/eMMC card bidirectional command/response signal.
SDMMC_CDIRE	Digital output	SD/SDIO/eMMC card I/O direction indication for the SDMMC_CMD signal.
SDMMC_D[7:0]	Digital input/output	SD/SDIO/eMMC card bidirectional data lines.
SDMMC_D0DIR	Digital output	SD/SDIO/eMMC card I/O direction indication for the SDMMC_D0 data line.
SDMMC_D123DIR	Digital output	SD/SDIO/eMMC card I/O direction indication for the data lines SDMMC_D[3:1].

### 48.4.3 General description

The **SDMMC\_D[7:0]** lines have different operating modes:

- By default, SDMMC\_D0 line is used for data transfer. After initialization, the host can change the databus width.
- For an eMMC, 1-bit (SDMMC\_D0), 4-bit (SDMMC\_D[3:0]) or 8-bit (SDMMC\_D[7:0]) data bus widths can be used.
- For an SD or an SDIO card, 1-bit (SDMMC\_D0) or 4-bit (SDMMC\_D[3:0]) can be used. All data lines operate in push-pull mode.

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the data lines is indicated with I/O direction signals. The **SDMMC\_D0DIR** signal indicates the I/O direction for the SDMMC\_D0 data line, the **SDMMC\_D123DIR** for the SDMMC\_D[3:1] data lines.

**SDMMC\_CMD** only operates in push-pull mode:

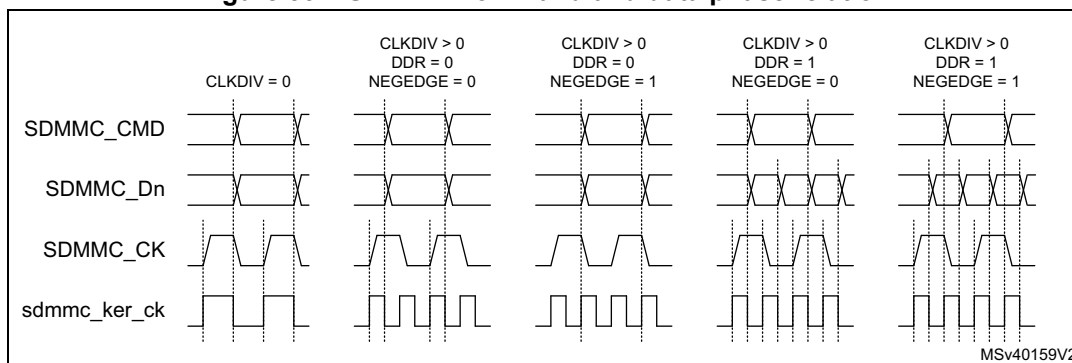
To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the SDMMC\_CMD line is indicated with the I/O direction signal **SDMMC\_CDIRE**.



**SDMMC\_CK** clock to the card originates from **sdmmc\_ker\_ck**:

- When the **sdmmc\_ker\_ck** clock has 50 % duty cycle, it can be used even in bypass mode (CLKDIV = 0).
- When the **sdmmc\_ker\_ck** duty cycle is not 50 %, the CLKDIV must be used to divide it by 2 or more (CLKDIV > 0).
- The phase relation between the SDMMC\_CMD / SDMMC\_D[7:0] outputs and the SDMMC\_CK can be selected through the NEGEDGE bit. The phase relation depends on the CLKDIV, NEGEDGE, and DDR settings. See [Figure 537](#).

**Figure 537. SDMMC Command and data phase relation**



**Table 294. SDMMC Command and data phase selection**

CLKDIV	DDR	NEGEDGE	SDMMC_CK	Command out	Data out
0	x	x	= sdmmc_ker_ck	generated on sdmmc_ker_ck falling edge	
>0	0	0	generated on sdmmc_ker_ck rising edge	generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.	
		1		generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.	
	1	0		generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.	generated on sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.
		1		generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.	

By default, the **sdmmc\_io\_in\_ck** feedback clock input is selected for sampling incoming data in the SDMMC receive path. It is derived from the SDMMC\_CK pin.

When using an external driver (a voltage switch transceiver), the SDMMC\_CKIN feedback clock input can be selected to sample the receive data.

For an SD/SDIO/eMMC card, the clock frequency can vary between 0 and 208 MHz (limited by maximum I/O speed).

Depending on the selected bus mode (SDR or DDR), one bit or two bits are transferred on SDMMC\_D[7:0] lines with each clock cycle. The SDMMC\_CMD line transfers only one bit per clock cycle.

#### 48.4.4 SDMMC adapter

The SDMMC adapter (see [Figure 536: SDMMC block diagram](#)) is a multimedia/secure digital memory card bus master that provides an interface to a MultiMediaCard stack or to a secure digital memory card. It consists of the following subunits:

- Control unit
- Data transmit path
- Command path
- Data receive path
- Response path
- Receive data path clock multiplexer
- Adapter register block
- Data FIFO
- Internal DMA (IDMA)

*Note:* The adapter registers and FIFO use the AHB clock domain (`sdmmc_hclk`). The control unit, command path and data transmit path use the SDMMC adapter clock domain (`sdmmc_ker_ck`). The response path and data receive path use the SDMMC adapter feedback clock domain from the `sdmmc_io_in_ck`, or `SDMMC_CKIN`.

##### Adapter register block

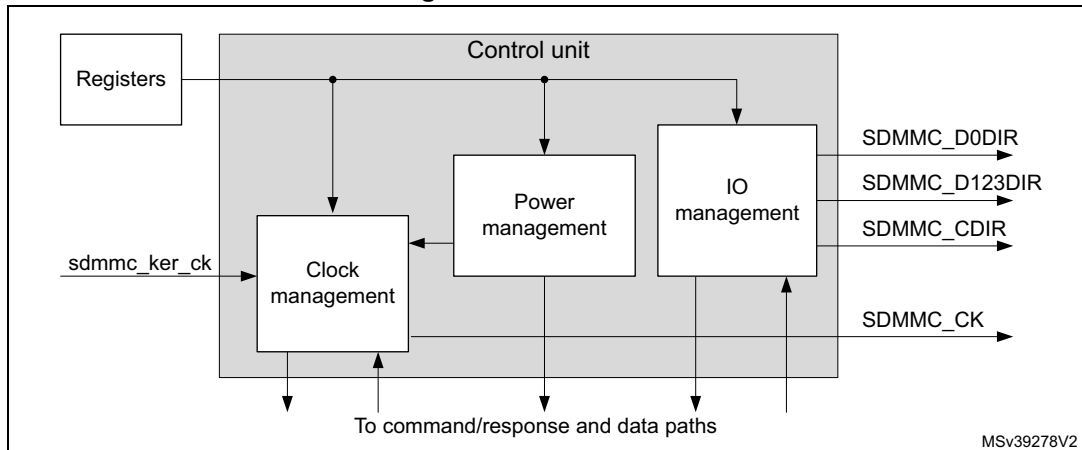
The adapter register block contains all system control registers, the SDMMC command and response registers and the data FIFO.

This block also generates the signals from the corresponding bit location in the SDMMC Clear register that clear the static flags in the SDMMC adapter.

##### Control unit

The control unit illustrated in [Figure 538](#), contains the power management functions, the SDMMC\_CK clock management with divider, and the I/O direction management.

Figure 538. Control unit



The power management subunit disables the card bus output signals during the power-off and power-up phases.

There are three power phases:

- power-off
- power-up
- power-on

The clock management subunit uses the `sdmmc_ker_ck` to generate the `SDMMC_CK` and provides the division control. It also takes care of stopping the `SDMMC_CK` for i.e. flow control.

The clock outputs are inactive:

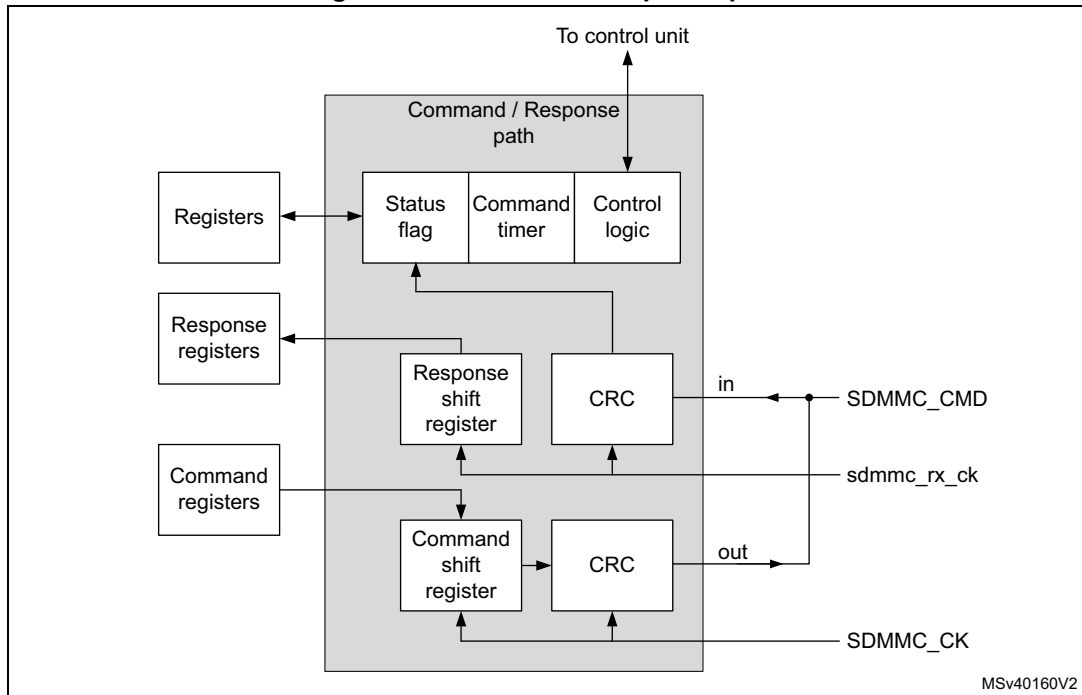
- after reset
- during the power-off or power-up phases
- if the power saving mode (register bit `PWRSV`) is enabled and the card bus is in the Idle state for eight clock periods. The clock will be stopped eight cycles after both the command/response CPSM and data path DPSM subunits have entered the Idle phase. The clock will be restarted when the command/response CPSM or data path DPSM is activated (enabled).

The I/O management subunit takes care of the `SDMMC_Dn` and `SDMMC_CMD` I/O direction signals, which controls the external voltage transceiver.

### Command/Response path

The Command/Response path subunit transfers commands and responses on the `SDMMC_CMD` line. The Command path is clocked on the `SDMMC_CK` and sends commands to the card. The Response path is clocked on the `sdmmc_rx_ck` and receives responses from the card.

Figure 539. Command/Response path

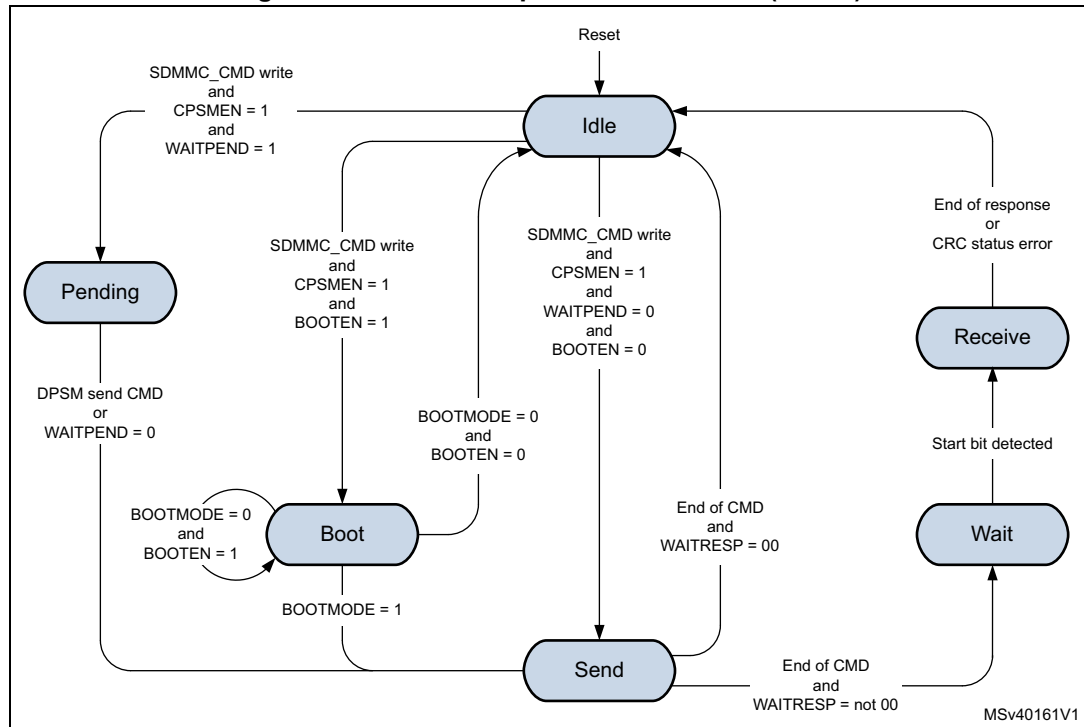


Command/Response path state machine (CPSM)

- When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent the CRC is appended and the command path state machine (CPSM) sets the status flags and:
  - if a response is not required enters the Idle state.
  - If a response is required, it waits for the response.
- When the response is received,
  - for a response with CRC, the received CRC code and the internally generated code are compared, and the appropriate status flag is set according the result.
  - for a response without CRC, no CRC is checked, and the appropriate status flag is not set.

When ever the CPSM is active, i.e. not in the Idle state, the CPSMACT bit is set.

Figure 540. Command path state machine (CPSM)



- **Idle:** The command path is inactive. When the command control register is written and the enable bit (CPSMEN) is set, the CPSM will activate the SDMMC\_CK clock (when stopped due to power save PWRSAPV bit) and moves
  - to the Send state when WAITPEND = 0 & BOOTEN = 0.
  - to the Pending state when WAITPEND = 1.
  - to the boot state when BOOTEN = 1.
- **Send:** The command is sent and the CRC is appended.
  - When CMDTRANS bit is set or when BOOTEN bit is set and BOOTMODE is alternative boot, and the DTDIR = receive, the CPSM DataEnable signal will be issued to the DPSM at the end of the command.
  - When the CMDTRANS bit is set and the CMDSUSPEND bit is 0 the interrupt period will be terminated at the end of the command.
  - When CMDSTOP bit is set the CPSM Abort signal will be issue to the DPSM at the end of the command.
  - If no response is expected (WAITRESP = 00) the CPSM will move to the Idle state and generate the CMDSENT flag. When BOOTMODE = 1 & BOOTEN = 0 the CMDSENT flag is delayed 56 cycles after the command End bit, otherwise the

- CMDSENT flag is generated immediately after the command End bit.  
The RESPCMDR and RESPxR registers are not modified.
- If a command response is expected (WAITRESP = not 00) the CPSM will move to the Wait state and start the response timeout.
  - **Wait:** The Command path waits for a response.
    - When WAITINT bit is 0 the command timer starts running and the CPSM waits for a Start bit.
      - a) If a Start bit is detected before the timeout the CPSM moves to the Receive state.
      - b) If the timeout is reached before the CPSM detect a response start bit, the timeout flag (CTIMEOUT) is set and the CPSM moves to the Idle state.  
The RESPCMDR and RESPxR registers are not modified.
    - When WAITINT bit is 1, the timer is disabled and the CPSM waits for an interrupt request (Response Start bit) from one of the cards.
      - a) When a Start bit is detected the CPSM moves to the Receive state.
      - b) When writing WAITINT to 0 (interrupt mode abort), the host will send a response by its self and on detecting the Start bit the CPSM move to the Receive state.
  - **Receive:** The command response will be received. Depending the response mode bits WAITRESP in the command control register, the response can be either short or long, with CRC or without CRC. The received CRC code when present will be verified against the internally generated CRC code.
    - When the CMDSUSPEND bit is set and the SDIO Response bit BS = 0 (response bit [39]), the interrupt period will be started after the response.  
When the CMDSUSPEND bit is cleared, or the CMDSUSPEND bit is 1 and the SDIO Response bit BS = 1 (response bit [39]), there will be no interrupt period started.
    - When the CMDTRANS bit is set and the CMDSUSPEND bit is set and the SDIO Response bit DF= 1 (response bit [32]) the interrupt period will be terminated after the response.
    - When the CRC status passes or no CRC is present the CMDREND flag is set, the CPSM moves to the Idle state.  
The RESPCMDR and RESPxR registers are updated with received response.
      - When BOOTMODE = 1 & BOOTEN = 0 the CMDREND flag is delayed 56 cycles after the response End bit, otherwise the CMDREND flag is generated immediately after the response End bit.
      - When CMDTRANS bit is set and the DTDIR = transmit, the CPSM DataEnable signal will be issued to the DPSM at the end of the command response.
    - When the CRC status fails the CCRCFAIL flag is set and the CPSM moves to the Idle state.  
The RESPCMDR and RESPxR registers are updated with received response.
  - **Pending:** According the pending WAITPEND bit in the command register, the CPSM enters the pending state.
    - When DATALENGTH =< 5 bytes the CPSM moves to the Sent state en generates the DataEnable signal to start the data transfer aligned with the CMD12 Stop Transmission command.
    - When DATALENGTH > 5 bytes, the CPSM DataEnable signal will be issued to the DPSM to start the data transfer. The CPSM waits for a sendCMD signal from the

DPSM before moving to the Sent state. This enables i.e. the CMD12 Stop Transmission command to be sent aligned with the data.

- When writing WAITPEND to 0, the CPSM will move to the Sent state.
- **Boot:** If the BOOTEN bit is set in the command register, the CPSM enters the boot state, and when:
  - BOOTMODE = 0 the SDMMC\_CMD line is driven low and when CMDTRANS bit is set and the DTDIR = receive, the CPSM DataEnable signal will be issued to the DPSM. This enables normal boot operation. This state is left at the end of the boot procedure by clearing the register bit BOOTEN, which cause the SDMMC\_CMD line to be driven high and the CPSM Abort signal will be issued to the DPSM, before moving to the Idle state. The CMDSENT flag is generated 56 cycles after SDMMC\_CMD line is high.
  - BOOTMODE = 1, move to the Send state. This enables sending of the CMD0 (boot). Clearing BOOTEN has no effect.

*Note:* The CPSM remains in the Idle state for at least eight SDMMC\_CK periods to meet the  $N_{CC}$  and  $N_{RC}$  timing constraints.  $N_{CC}$  is the minimum delay between two host commands, and  $N_{RC}$  is the minimum delay between the host command and the card response.

*Note:* The response timeout has a fixed value of 64 SDMMC\_CK clock periods.

A Command is a token that starts an operation. Commands are sent from the host to either a single card (addressed command) or all connected cards (broadcast command are available for eMMC V3.31 or previous). Commands are transferred serially on the SDMMC\_CMD line. All commands have a fixed length of 48 bits. The general format for a command token for SD-Memory cards, SDIO cards, and eMMC cards is shown in [Table 295](#).

The Command token data is taken from 2 registers, one containing a 32-bits argument and the other containing the 6-bits command index (six bits sent to a card).

**Table 295. Command token format**

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	1	Transmission bit
[45:40]	6	x	Command index
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

Next to the Command data there are command type (WAITRESP) bits controlling the command path state machine (CPSM). These bits also determine whether the command requires a response, and whether the response is short (48 bit) or long (136 bits) long, and if a CRC is present or not.

A Response is a token that is sent from an addressed card or synchronously from all connected cards to the host as an answer to a previous received Command. All responses are sent via the command line SDMMC\_CMD. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The code length depends on the response type. Response tokens R1, R2, R3, R4, R5, and R6 have various

coding schemes, depending on their content. The general formats for the response tokens for SD-Memory cards, SDIO cards, and eMMC cards are shown in [Table 296](#), [Table 297](#) and [Table 298](#).

A response always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (card = 0). A value denoted by x in the tables below indicates a variable entry. Most responses, except some, are protected by a CRC. Every command code word is terminated by the End bit (always 1).

The Response token data is stored in 5 registers, four containing the 32-bits card status, OCR register, argument or 127-bits CID or CSD register including internal CRC, and one register containing the 6-bits command index.

**Table 296. Short response with CRC token format**

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

**Table 297. Short response without CRC token format**

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	1111111	(reserved 1111111)
0	1	1	End bit

**Table 298. Long response with CRC token format**

Bit position	Width	Value	Description
135	1	0	Start bit
134	1	0	Transmission bit
[133:128]	6	111111	Reserved
[127:1]	127:8	x	CID or CSD slices
	7:1	x	CRC7 (included in CID or CSD)
0	1	1	End bit

The Command/Response path operates in a half-duplex mode, so that either commands can be sent or responses can be received. If the CPSM is not in the Send state, the



SDMMC\_CMD output is in the Hi-Z state. Data sent on SDMMC\_CMD are synchronous with the SDMMC\_CK according to the NEGEDGE register bit see [Figure 537](#).

The Command and Short Response with CRC, the CRC generator calculates the CRC checksum for all 40 bits before the CRC code. This includes the start bit, transmitter bit, command index, and command argument (or card status).

For the Long Response the CRC checksum is calculated only over the 120 bits of R2 CID or CSD. Note that the start bit, transmitter bit and the six reserved bits are not used in the CRC calculation.

The CRC checksum is a 7-bit value:

$$\text{CRC}[6:0] = \text{Remainder} [(M(x) * x^7) / G(x)]$$

$$G(x) = x^7 + x^3 + 1$$

$$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + \dots + (\text{last bit before CRC}) * x^0$$

Where n = 39 or 119.

The CPSM allows to send a number of specific commands to handle various operating modes when CPSMEN is set, see [Table 299](#).

**Table 299. Specific Commands overview**

VSWITCH	BOOTEN	BOOTMODE	CMDTRANS	WAITPEND	CMDSTOP	WAITINT	Description
1	x	x	x	x	x	x	Start Voltage Switch Sequence
0	1	x	x	x	x	x	Start normal boot
0	1	1	x	x	x	x	Start alternative boot
0	0	1	x	x	x	x	Stop alternative boot.
0	0	0	1	x	x	x	Send command with associated data transfer.
0	0	0	0	1	1	x	eMMC stream data transfer, command (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	1	0	x	eMMC stream data transfer, command different from (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	0	1	x	Send command (STOP_TRANSMISSION), stopping any ongoing data transmission.
0	0	0	0	0	0	1	Enter eMMC wait interrupt (Wait-IRQ) mode.
0	0	0	0	0	0	0	Any other none specific command

The Command/Response path implements the status flags and associated clear bits shown in [Table 300](#):

**Table 300. Command path status flags**

Flag	Description
CMDSENT	Set at the end of the command without response. (CPSM moves from SEND to IDLE)
CMDREND	Set at the end of the command response when the CRC is OK. (CPSM moves from RECEIVE to IDLE)
CCRCFAIL	Set at the end of the command response when the CRC is FAIL. (CPSM moves from RECEIVE to IDLE)
CTIMEOUT	Set after the command when no response start bit received before the timeout. (CPSM moves from WAIT to IDLE)
CKSTOP	Set after the voltage switch (VSWITCHEN = 1) command response when the CRC is OK and the SDMMC_CK is stopped. (no impact on CPSM)
VSWEND	Set after the voltage switch (VSWITCH = 1) timeout of 5ms + 1ms. (no impact on CPSM)
CPSMACT	Command transfer in progress. (CPSM not in Idle state)

The Command path error handling is shown in [Table 304](#):

**Table 301. Command path error handling**

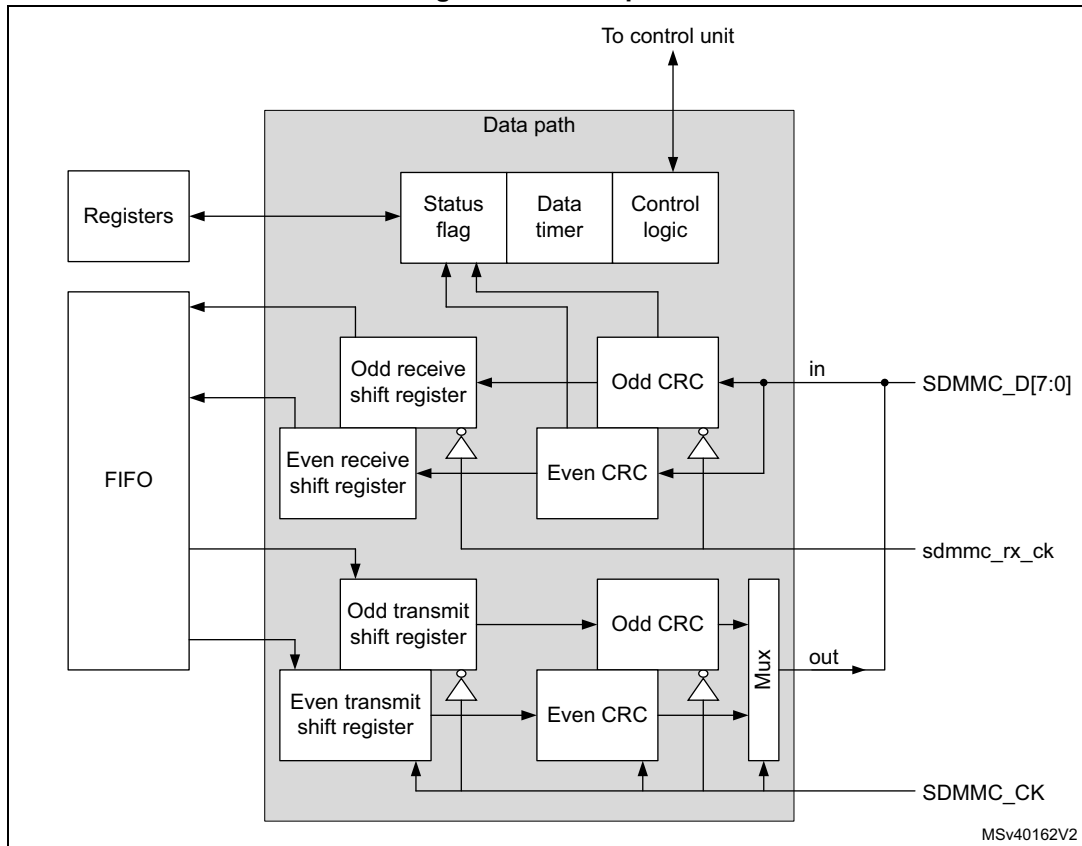
Error	CPSM state	Cause	Card action	Host action	CPSM action
Timeout	Wait	No start bit in time	Unknown	Reset or cycle power card <sup>(1)</sup>	Move to Idle
CRC status	Receive	Negative status	Command ignored	Resend command <sup>(1)</sup>	Move to Idle
		Transmission error	Command accepted	Resend command <sup>(1)</sup>	

1. When CMDTRANS is set, also a stop\_transmission command shall be send to move the DPSM to Idle.

## Data path

The data path subunit transfers data on the SDMMC\_D[7:0] lines to and from cards. The data transmit path is clocked on the SDMMC\_CK and sends data to the card. The data receive path is clocked on the sdmmc\_rx\_ck and receives data from the card. [Figure 541](#) shows the data path block diagram.

Figure 541. Data path



MSv40162V2

The card data bus width can be programmed in the clock control register bits WIDBUS. The supported data bus width modes are:

- If the wide bus mode is not enabled, only one bit is transferred over SDMMC\_D0.
- If the 4-bit wide bus mode is enabled, data is transferred at four bits over SDMMC\_D[3:0].
- If the 8-bit wide bus mode is enabled, data is transferred at eight bits over SDMMC\_D[7:0].

Next to the data bus width the data sampling mode can be programmed in the clock control register bit DDR. The supported data sampling modes are:

- Single data rate signaling (SDR), data is clocked on the rising edge of the clock.
- Double data rate signaling (DDR), data is clocked on the both edges of the clock. DDR mode is only supported in wide bus mode (4-bit wide and 8-bit wide).

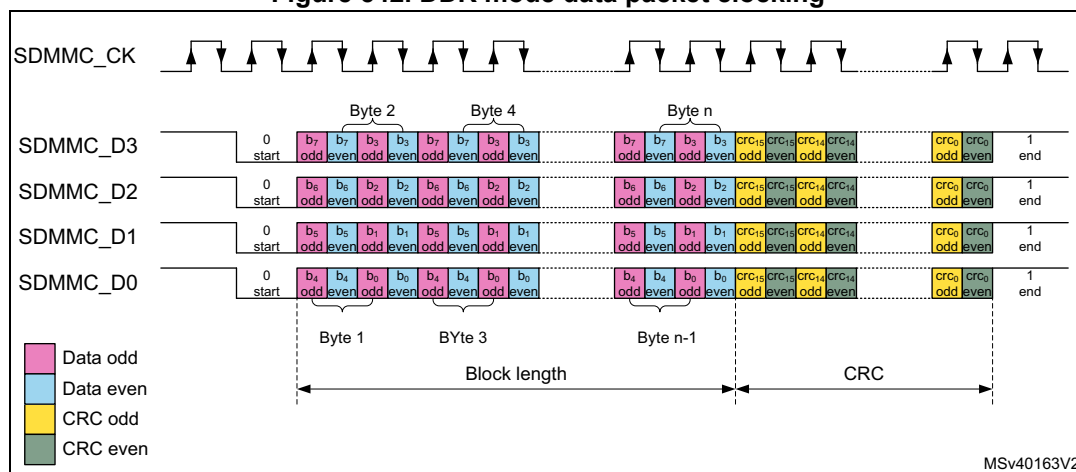
*Note: The data sampling mode only applies to the SDMMC\_D[7:0] lines. (not applicable to the SDMMC\_CMD line.)*

In DDR mode, data is sampled on both edges of the SDMMC\_CK according the following rules, see also *Figure 542* and *Figure 543*:

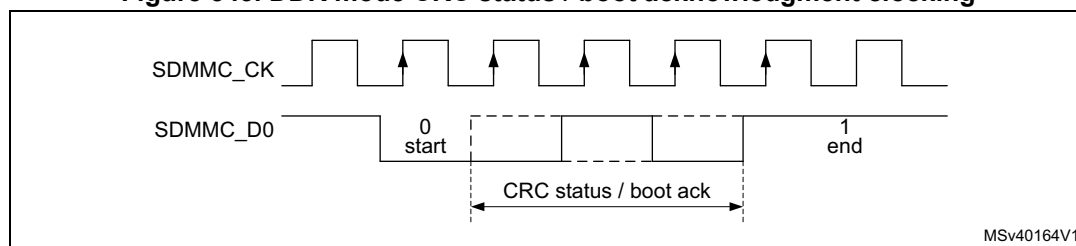
- On the rising edge of the clock Odd bytes are sampled.
- On the falling edge of the clock Even bytes are sampled.
- Data payload size is always a multiple of 2 Bytes.
- Two CRC16 are computed per data line
  - Odd bits CRC16 clocked on the falling edge of the clock.
  - Even bits CRC16 clocked on the rising edge of the clock.
- Start, End bits and idle conditions are full cycle.
- CRC status / boot acknowledgment and Busy signaling are full cycle and are only sampled on the rising edge of the clock.

In DDR mode the SDMMC\_CK clock division shall be  $\geq 2$ .

**Figure 542. DDR mode data packet clocking**



**Figure 543. DDR mode CRC status / boot acknowledgment clocking**



Data path state machine (DPSM)

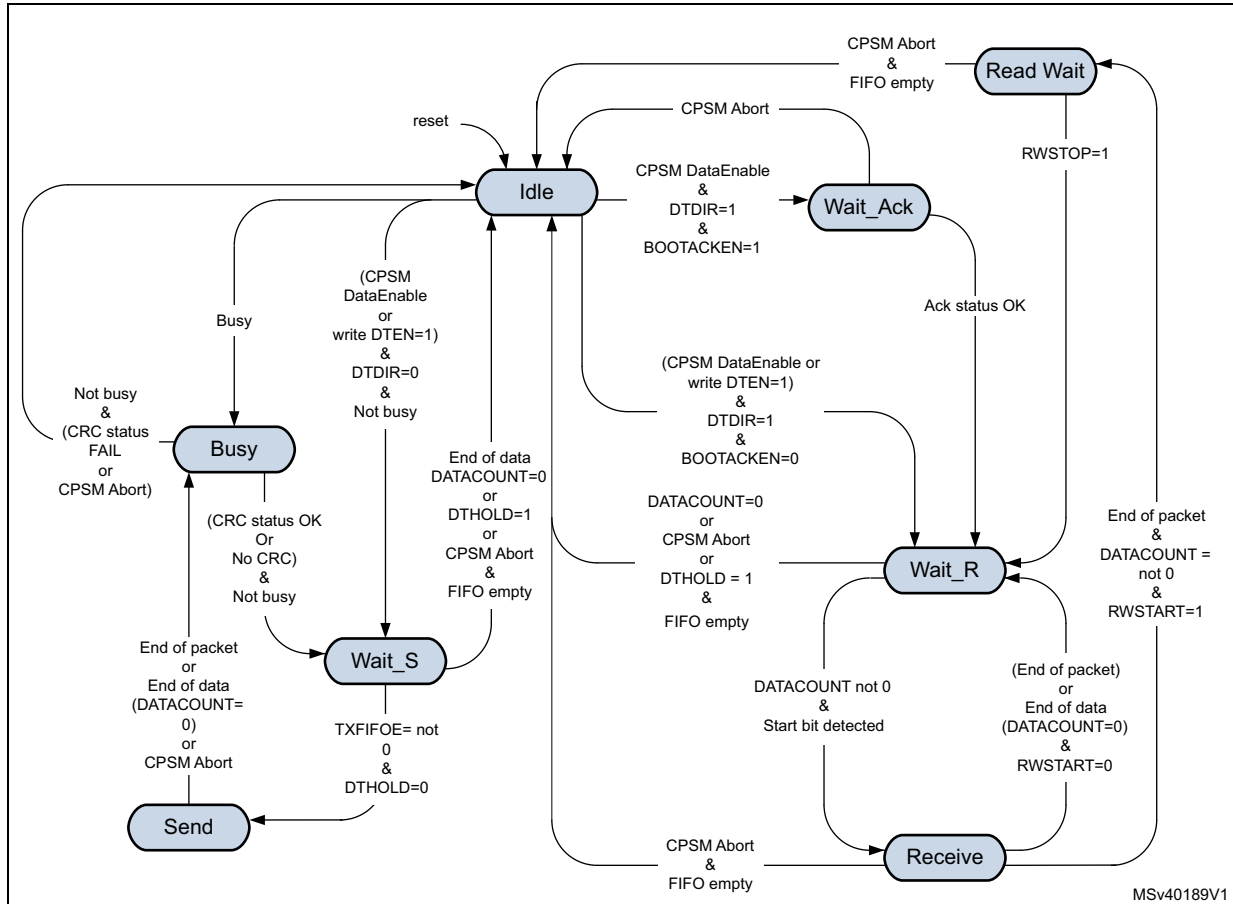
Depending on the transfer direction (send or receive), the data path state machine (DPSM) moves to the Wait\_S or Wait\_R state when it is enabled:

- Send: the DPSM moves to the Wait\_S state. If there is data in the transmit FIFO, the DPSM moves to the Send state, and the data path subunit starts sending data to a card.
- Receive: the DPSM moves to the Wait\_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the Receive state, and the data path subunit starts receiving data from a card.

For boot operation with acknowledgment the DPSM moves to the Wait\_Ack state and waits for the boot acknowledgment before moving to the Wait\_R state.

The DPSM operates at SDMMC\_CK. The DPSM has the following states, as shown in Figure 544. When ever the DPSM is active, i.e. not in the Idle state, the DPSMACT bit is set.

Figure 544. Data path state machine (DPSM)



- Idle state:** the data path is inactive, and the SDMMC\_D[7:0] outputs are according to the PWRCTRL setting. The DPSM will be activated either by sending a command with CMDTRANS bit set or by setting the DTEN bit, or by detecting Busy on SDMMC\_D0 (that is, after a command with R1b response).

When not busy, the DPSM will activate the SDMMC\_CK clock (when stopped due to power save PWRSAV bit), loads the data counter with a new (DATALENGTH) value and:

- When the data direction bit (DTDIR) indicates send, moves to the Wait\_S.
- When the data direction bit (DTDIR) indicates receive, moves to the
  - Wait\_R when BOOTACKEN register bit is clear.
  - Wait\_Ack when BOOTACKEN register bit is set and start the acknowledgment timeout.

When busy the DPSM will keep the SDMMC\_CK clock active and move to the Busy state.

Note: *DTEN shall not be used to start data transfer with SD, SDIO and eMMC cards.*

- **Wait\_Ack** state: the data path waits for the boot acknowledgment token.
  - The DPSM moves to the Wait\_R state if it receives an error free acknowledgment before a timeout.
  - When a pattern different from the acknowledgment is received an acknowledgment status error is generated, and the Ack fail status flag (ACKFAIL) is set. The DPSM stays in Wait\_Ack.
  - If it reaches a timeout (ACKTIME) before it detects a start bit, it sets the timeout status flag (ACKTIMEOUT). The DPSM stays in Wait\_Ack.
  - When the CPSM Abort signal is set it moves to the Idle state and sets the DABORT flag.
- **Wait\_R** state: the data path, if the data counter is not zero and data is not hold, waits for a start bit on SDMMC\_D[n:0]. If the data counter is zero or data is hold, wait for the FIFO to be empty.
  - In block mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DBLOCKSIZE.
  - In SDIO multibyte mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DATALENGTH.
  - In stream mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data counter with DATALENGTH.
  - if the data counter (DATACOUNT) equals zero (end of data) the DPSM moves to the Idle state when the receive FIFO is empty and the DATAEND flag is set.
  - If it reaches a timeout (DATETIME) before it detects a start bit, it sets the timeout status flag (DTIMEOUT) and the DPSM stays in the WAIT\_R state.
  - If the CPSM Abort signal is set:
    - If DATACOUNT > 0, the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST, and sets the DABORT flag.
    - If DATACOUNT is zero normal operation is continued, there will be no DABORT flag since the transfer has completed normally.
  - if the DTHOLD bit is set:
    - When DATACOUNT > 0, the DPSM moves to the Idle state when the receive FIFO is empty and when IDMAEN = 0 reset with FIFORST, and issues the DHOLD flag. When Holding the timeout is disabled. When an CPSM Abort signal is received during Holding, the transfer is Aborted.

- When DATACOUNT = 0, the transfer is completed normally and there will be no DHOLD flag.
- When DPSM has been started with DTEN, after an error (DTIMEOUT) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.
- **ReadWait** state: the data path ReadWait the bus.
  - The DPSM moves to the Wait\_R state when the ReadWait stop bit (RWSTOP) is set, and start the receive timeout.
  - If the CPSM Abort signal is set, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then moves to the Idle state and sets the DABORT flag.
- **Receive** state: the data path receives serial data from a card. Pack the data in bytes and written it to the data FIFO. Depending on the transfer mode selected in the data control register (DTMODE), the data transfer mode can be either block or stream:
  - In block mode, when the data block size (DBLOCKSIZE) number of data bytes are received, the DPSM waits until it receives the CRC code.
  - In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are received, the DPSM waits until it receives the CRC code.
  - a) If the received CRC code matches the internally generated CRC code, the DPSM moves to the
    - Wait\_R state when RWSTART= 0 and start the receive timeout.
    - ReadWait state when RWSTART = 1 and DATACOUNT > zero, and generate the DBCKEND flag.
  - b) If the received CRC code fails the internally generated CRC code any further data reception is prevented.
    - When not all data has been received (DATACOUNT > 0), the CRC fail status flag (DCRCFAIL) is set and the DPSM stays in the Receive state.
    - When all data has been received (DATACOUNT = 0), wait for the FIFO to be empty after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  - In stream mode, the DPSM receives data while the data counter DATACOUNT > 0. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DPSM moves to the Wait\_R state.
  - When a FIFO overrun error occurs, the DPSM sets the FIFO overrun error flag (RXOVERR) and any further data reception is prevented. The DPSM stays in the Receive state.
  - When an CPSM\_Abort signal is received:
    - If the CPSM\_Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted. The remaining data in the shift register is written to the data FIFO, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and the DABORT flag is set.
    - If the CPSM\_Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT=0, the transfer is completed normally. The DPSM stays in the Receive state no DABORT flag is generated.
  - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or RXOVERR) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.

- **Wait\_S** state: the data path waits for data to be available from the FIFO.
  - If the data counter `DATACOUNT` > 0, waits until the data FIFO empty flag (`TXFIFOE`) is de-asserted and `DTHOLD` is not set, and moves to the Send state.
  - if the data counter (`DATACOUNT`) = 0 the `DPSM` moves to the Idle state and generate the `DATAEND` flag.
  - When `DTHOLD` is set and the `DATACOUNT` > 0
    - When `IDMA` is enabled, the FIFO will be flushed, then the `DPSM` will move to the Idle state and issues the `DHOLD` flag. (The `DBCKEND` flag will also be set, and can be cleared at the same time as the `DHOLD` flag.)
    - When `IDMA` is disabled generate the `DBCKEND` flag. wait for the FIFO to be reset with `FIFORST`, then `DPSM` will move to the Idle state and issues the `DHOLD` flag.
  - When `DTHOLD` is set and `DATACOUNT` = 0 the transfer is completed normally.
  - When receiving the `CPSM Abort` signal
    - If the `CPSM_Abort` signal is received before the 2 last bits of the data with `DATACOUNT` = 0, the transfer is aborted, wait for the FIFO to be empty and when `IDMAEN` = 0 reset with `FIFORST`, then the `DPSM` moves to the Idle state and sets the `DABORT` flag.
    - If the `CPSM_Abort` signal is received during or after the 2 last bits of the transfer with `DATACOUNT`=0, normal operation is continued, there will be no `DABORT` flag since the transfer has completed normally.

*Note:* The `DPSM` remains in the `Wait_S` state for at least two clock periods to meet the  $N_{WR}$  timing requirements, where  $N_{WR}$  is the number of clock cycles between the reception of the card response and the start of the data transfer from the host.

- **Send** state: the `DPSM` starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block, `SDIO` multibyte or stream:
  - In block mode, when the data block size (`DBLOCKSIZE`) number of data bytes are send, the `DPSM` sends an internally generated CRC code and End bit, and moves to the Busy state and start the transmit timeout.
  - In `SDIO` multibyte mode, when the data block size (`DATALENGTH`) number of data bytes are send, the `DPSM` sends an internally generated CRC code and End bit, and moves to the Busy state and start the transmit timeout.
  - In stream mode, the `DPSM` sends data to a card while the data counter `DATACOUNT` > 0. When the data counter reaches zero moves to the Busy state and start the transmit timeout.  
Before sending the last stream Byte according to `DATACOUNT`, the `DPSM` issues a trigger on the `sendCMD` signal. This signal is used by the `CPSM` to sent any pending command. (i.e. `CMD12 Stop Transmission` command)
  - If a FIFO underrun error occurs, the `DPSM` sets the FIFO underrun error flag (`TXUNDERR`). The `DPSM` stays in the Send state.
  - When receiving the `CPSM Abort` signal
    - If the `CPSM_Abort` signal is received before the 2 last bits of the transfer with `DATACOUNT`=0, the transfer is aborted. The `DPSM` will sent a last data bit followed by an End bit. The FIFO will be disabled/flushed, and the `DPSM` moves to the Busy state to wait for not busy before setting the `DABORT` flag.
    - If the `CPSM_Abort` signal is received during or after the 2 last bits of the transfer



with DATACOUNT=0, the transfer is completed normally, there will be no DABORT flag.

- **Busy state:** the DPSM waits for the CRC status token when expected, and wait for a not busy signal:
  - If a CRC status token is expected and indicate “non-erroneous transmission” or when there is no CRC expected:
    - it moves to the Wait\_S state when SDMMC\_D0 is not low (the card is not busy).
    - When the card is busy SDMMC\_D0 is low it will remain in the Busy state.
  - If a CRC status token is expected and indicates “erroneous transmission”.
    - When not all data has been send (DATACOUNT > 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set. The FIFO will be disabled/flushed and the DPSM stays in the Busy state.
    - When all data has been send (DATACOUNT = 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  - If a CRC status (Ncrc) timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - If a busy timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - When receiving the CPSM Abort signal in the Busy state:
    - If the CPSM\_Abort signal is received before the 2 last bits of the CRC response with DATACOUNT > 0, the data transfer is aborted. The DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
    - If the CPSM\_Abort signal is received during or after the 2 last bits of the CRC response when DATACOUNT=0 or when no CRC is expected and DATACOUNT = 0 and there has been no DTIMEOUT error, the DPSM stays in the Busy state no DABORT flag is generated, since the transfer may completed normally.
    - If the CPSM\_Abort signal is received when a DTIMEOUT error has occurred the DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
  - When entering the Busy state due to an Abort in the Send state, the DPSM waits for not busy before moving to the Idle state and the DABORT flag is set.
  - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or DTIMEOUT) the DPSM moves to the Idle state when the FIFO is reset.
  - When the DPSM has been started due to Busy on SDMMC\_D0, waits for not busy after which the Busy end status flag (BUSYD0END) is set and the DPSM moves to the Idle state.

The data timer (DATATIME) is enabled when the DPSM is in the Wait\_R or Busy state 2 cycles after the data block end bit, or data read command end bit, or R1b response, and generates the data timeout error (DTIMEOUT):

- When transmitting data, the timeout occurs
  - when a CRC status is expected and no start bit is received within 8 SDMMC\_CK cycles, the DTIMEOUT flag is set.
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.
- When receiving data, the timeout occurs
  - when there is still data to be received DATACOUNT > 0 and no start bit is received before the programmed timeout period, the DTIMEOUT flag is set.
- After a R1b response, the timeout occurs
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.

When DATATIME = 0,

- In receive the start bit shall be present 2 cycles after the data block end bit or data read command end bit.
- In transmit busy is timed out 2 cycles after the CRC token end bit or stream data end bit.
- After a R1b response busy is timed out 2 cycles after the response end bit.

Data can be transferred from the card to the host (transmit, send) or vice versa (receive). Data are transferred via the SDMMC\_Dn data lines, they are stored in a FIFO.

**Table 302. Data token format**

Description	Start bit	Data <sup>(1)</sup>	CRC16	End bit	DTMODE
Block data	0	(DBLOCKSIZE, DATALENGTH)	yes	1	00
SDIO multibyte	0	(DATALENGTH)	yes	1	01
eMMC stream	0	(DATALENGTH)	no	1	10

1. The total amount of data to transfer is given by DATALENGTH. Where for Block data the amount of data in each block is given by DBLOCKSIZE.

The data token format is selected with register bits DTMODE according.

The data path implements the status flags and associated clear bits shown in [Table 303](#):

**Table 303. Data path status flags and clear bits**

Flag		Description
DATAEND	TX	Set at the end of the complete data transfer when the CRC is OK and busy has finished. (DATACOUNT = 0). (DPSM moves from WAIT_S to IDLE)
	RX	Set at the end of the complete data transfer when the CRC is OK and all data has been read. (DATACOUNT = 0 and FIFO is empty). (DPSM moves from WAIT_R to IDLE)
	BOOT	

Table 303. Data path status flags and clear bits (continued)

Flag		Description
DCRCFAIL	TX	Set at the end of the CRC when FAIL and busy has finished. (DPSM stay in BUSY when there is still data to send and wait for Abort) (DPSM moves from BUSY to IDLE when all data has been sent) or DPSM has been started with DTEN
	RX	Set at the end of the CRC when FAIL and FIFO is empty. (DPSM stays in RECEIVE when there is still data to be received and wait for Abort) (DPSM moves from RECEIVE to IDLE when all data has been received or DPSM has been started with DTEN)
	BOOT	
ACKFAIL	BOOT	Set at the end of the BOOT ACK when FAIL. (DPSM stays in Wait_Ack and wait for Abort)
DTIMEOUT	CMD R1b	Set after the command response no end of busy received before the timeout. (DPSM stays in BUSY and wait for Abort)
	TX	Set when no CRC token start bit received within Ncrc, or no end of busy received before the timeout. (DPSM stays in BUSY and wait for Abort) (When DPSM has been started with DTEN move to IDLE) Note: The DCRCFAIL flag may also be set when CRC failed before the busy timeout.
	RX	Set when no start bit received before the timeout. (DPSM stays in WAIT_R and wait for Abort) (When DPSM has been started with DTEN move to IDLE)
	BOOT	
ACKTIMEOUT	BOOT	Set when no start bit received before the timeout. (DPSM stays in Wait_Ack and wait for Abort)
DBCKEND	TX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and busy has finished, when data transfer is not complete (DATACOUNT > 0). (DPSM moves from WAIT_S to IDLE)
	RX	When RWSTART = 1: Set at the end of data block transfer when the CRC is OK, when data transfer is not complete (DATACOUNT > 0). (DPSM moves from RECEIVE to READWAIT)
	BOOT	
DHOLD	TX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and busy has finished, when data transfer is not complete (DATACOUNT > 0). (DPSM moves from WAIT_S to IDLE)
	RX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and all data has been read (FIFO is empty), when data transfer is not complete (DATACOUNT > 0). (DPSM moves from WAIT_R to IDLE)
DABORT	CMD R1b	When Abort event has been sent by the CPSM and busy has finished. (DPSM moves from BUSY to IDLE)
	TX	When Abort event has been sent by the CPSM before the 2 last bits of the transfer. (DPSM moves from Any state to IDLE)
	RX	
	BOOT	
BUSYD0END	CMD R1b	Set after the command response when end of busy before the timeout. (DPSM moves from BUSY to IDLE)
DPSMACT		Data transfer in progress. (DPSM not in Idle state)

The data path error handling is shown in [Table 304](#):

Table 304. Data path error handling

Error	DPSM state	Cause	Card action	Host action	DPSM action
Timeout	Wait_Ack	No Ack in time	unknown	Card cycle power	Stay in Wait_Ack (reset the SDMMC with the RCC.SDMMCxRST register bit)
	Wait_R	No Start bit in time	unknown	Stop data reception Send stop transmission command	On CPSM_Abort move to Idle
			unknown	Stop boot procedure	
	Busy	Busy too long (due to data transfer)	unknown	Stop data reception Send stop transmission command	
Busy too long (due to R1b)		unknown	Send reset command		
CRC	Receive	transmission error	Send further data	Stop data reception Send stop transmission command	On CPSM_Abort move to Idle
CRC status	Busy	Negative status	Ignore further data	Stop data transmission Send stop transmission command	On CPSM_Abort move to Idle
		transmission error	wait for further data		
Ack status	Wait_Ack	transmission error	Send boot data	Stop boot procedure	On CPSM_Abort move to Idle
Overrun	Receive	FIFO full	Send further data	Stop data reception Send stop transmission command	On CPSM_Abort move to Idle
Underrun	Send	FIFO empty	Receive further data	Stop data transmission Send stop transmission command	On CPSM_Abort move to Idle

### Data FIFO

The data FIFO (first-in-first-out) subunit contains the transmit and receive data buffer. A single FIFO is used for either transmit or receive as selected by the DTDIR bit. The FIFO contain a 32-bit wide, 16-word deep data buffer and control logic. Because the data FIFO operates in the AHB clock domain (sdmmc\_hclk), all signals from the subunits in the SDMMC clock domain (SDMMC\_CK/sdmmc\_rx\_ck) are resynchronized.

The FIFO can be in one of the following states:

- The transmit FIFO refers to the transmit logic and data buffer when sending data out to the card. (DTDIR = 0)
- The receive FIFO refers to the receive logic and data buffer when receiving data in from the card. (DTDIR = 1)

The end of a correctly completed SDMMC data transfer from the FIFO is indicated by the DATAEND flags driven by the data path subunit. Any incorrect (aborted) SDMMC data transfer from the FIFO is indicated by one of the error flags (DCRCFAIL, DTIMEOUT, DABORT) driven by the data path subunit, or one of the FIFO error flags (TXUNDERR, RXOVERR) driven by the FIFO control.

The data FIFO can be accessed in the following ways, see [Table 305](#).

**Table 305. Data FIFO access**

Data FIFO access	IDMAEN
From FW via AHB slave interface	0
From IDMA via AHB master interface	1

Transmit FIFO:

Data can be written to the transmit FIFO when the DPSM has been activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by FW via the AHB slave interface. The transmit FIFO is accessible via sequential addresses. The transmit FIFO contains a data output register that holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, it increments the read pointer and drives new data out. The transmit FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
  - For block data transfer (DTMODE = 0), DATALENGTH shall be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in transmit mode (DTDIR = 0).
  - Configures the FIFO in transmit mode.
3. Enabled the data transfer
  - either by sending a Command from the CPSM with the CMDTRANS bit set
  - or by setting DTEN bit
4. When (DPSMACT = 1) write data to the FIFO.
  - The DPSM will stay in the Wait\_S state until FIFO is full (TXFIFO = 1), or the number indicated by DATALENGTH.
  - The SDMMC start sending data as long as FIFO is not empty.
5. When the FIFO is half empty (TXFIFOHE flag), write data to the FIFO until FIFO is full (TXFIFO = 1), or last data has been written.
6. When last data has been written wait for end of data (DATAEND flag)
  - SDMMC has completely sent all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer error or transfer hold when IDMAEN = 0, FW shall stop writing to the FIFO and flush and reset the FIFO with the FIFORST register bit.

The transmit FIFO status flags are listed in [Table 306](#).

Table 306. Transmit FIFO status flags

Flag	Description
TXFIFO	Set to high when all transmit FIFO words contain valid data.
TXFIFOE	Set to high when the transmit FIFO does not contain valid data.
TXFIFOHE	Set to high when half or more transmit FIFO words are empty.
TXUNDERR	Set to high when an underrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

Receive FIFO:

Data can be read from the receive FIFO when the DPSM is activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by FW via the AHB slave interface. When the data path subunit receives a word of data, it drives the data on the write databus. The write pointer is incremented after the write operation completes. On the read side, the contents of the FIFO word pointed to by the current value of the read pointer is driven onto the read databus. The receive FIFO is accessible via sequential addresses. The receive FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
  - For block data transfer (DTMODE = 0), DATALENGTH shall be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in receive mode (DTDIR = 1).
  - Configures the FIFO in receive mode.
3. Enable the DPSM transfer
  - either by sending a command from the CPSM with the CMDTRANS bit set
  - or by setting DTEN bit.
4. When (DPSMACT = 1) the FIFO is ready to receive data.
  - The DPSM will write the received data to the FIFO.
5. When the FIFO is half full (RXFIFOHF flag), read data from the FIFO until FIFO is empty (RXFIFOE = 1).
6. When last data has been received end of data (DATAEND flag), read data from the FIFO until FIFO is empty (RXFIFOE = 1).
  - SDMMC has completely received all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer hold when IDMAEN = 0, FW shall read the remaining data until the FIFO is empty and reset the FIFO with the FIFORST register bit. This will cause the DPSM to go to the Idle state (DPSMACT = 0).

In case of a data transfer error when IDMAEN = 0, FW shall stop reading the FIFO and flush and reset the FIFO with the FIFORST register bit. This will cause the DPSM to go to the Idle state (DPSMACT = 0).

The receive FIFO status flags are listed in [Table 307](#).

**Table 307. Receive FIFO status flags**

Flag	Description
RXFIFO	Set to high when all receive FIFO words contain valid data
RXFIFOE	Set to high when the receive FIFO does not contain valid data.
RXFIFOHF	Set to high when half or more receive FIFO words contain valid data.
RXOVERR	Set to high when an overrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

**CLKMUX unit**

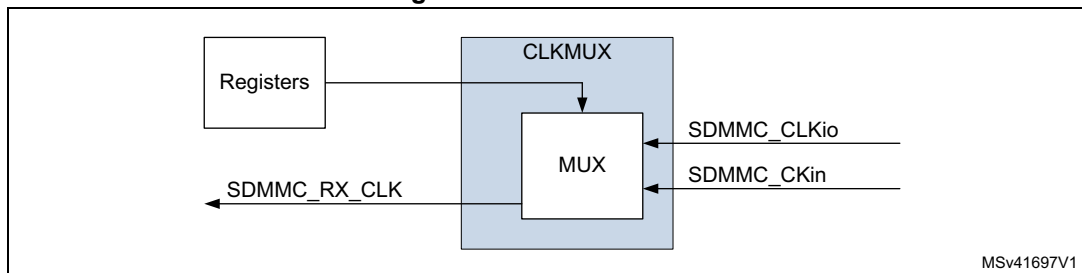
The CLKMUX selects the source for clock `sdmmc_rx_ck` to be used with the received data and command response. The receive data clock source can be selected by the clock control register bit `SELCLKRX`, between:

- `sdmmc_io_in_ck` bus master main feedback clock.
- `SDMMC_CKIN` external bus feedback clock.

The `sdmmc_io_in_ck` is selected when there is no external driver, with DS and HS.

The `SDMMC_CKIN` is selected when there is an external driver with SDR12, SDR25, SDR50 and DDR50.

**Figure 545. CLKMUX unit**



The `sdmmc_rx_ck` source shall be changed when the CPSM and DPSM are in the Idle state.

**48.4.5 SDMMC AHB slave interface**

The AHB slave interface generates the interrupt requests, and accesses the SDMMC adapter registers and the data FIFO. It consists of a data path, register decoder, and interrupt logic.

**SDMMC FIFO**

The FIFO access is restricted to word access:

- In transmit FIFO mode
  - Data are written to the FIFO in words (32-bits) until all data according `DATALENGTH` has been transferred. When the `DATALENGTH` is not an integer

multiple of 4, the last remaining data (1, 2 or 3 bytes) are written with a word transfer.

- In receive FIFO mode
  - Data are read from the FIFO in words (32-bits) until all data according DATALENGTH has been transferred. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are read with a word transfer padded with 0 value bytes.

When accessing the FIFO with half word or byte accesses an AHB bus fault is generated.

### SDMMC interrupts

The interrupt logic generates an interrupt request signal that is asserted when at least one of the unmasked status flags is active. A mask register is provided to allow selection of the conditions that will generate an interrupt. A status flag generates the interrupt request if a corresponding mask flag is set. Some status flags require an implicit clear in the clear register.

## 48.4.6 SDMMC AHB master interface

The AHB master interface is used to transfer the data between a memory and the FIFO using the SDMMC IDMA.

### SDMMC IDMA

Direct memory access (DMA) is used to provide high-speed transfer between the SDMMC FIFO and the memory. The AHB master optimizes the bandwidth of the system bus. The SDMMC internal DMA (IDMA) provides one channel to be used either for transmit or receive.

The IDMA is enabled by the IDMAEN bit and supports burst transfers of 8 beats.

- In transmit burst transfer mode:
  - Data are fetched in burst from memory whenever the FIFO is empty for the number of burst transfers, until all data according DATALENGTH has been transferred. When the DATALENGTH is not an integer multiple of the burst size the remaining, smaller than burst size data is transferred using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are fetched with a word transfer.
- In receive burst transfer mode:
  - Data are stored in burst in to memory whenever the FIFO contains the number of burst transfers, until all data according DATALENGTH has been transferred. When the DATALENGTH is not an integer multiple of the burst transfer the remaining, smaller than burst size data, is transferred using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are stored with halfword and or byte transfers.

In addition the IDMA provides the following channel configurations selected by bit IDMABMODE:

- single buffered channel
- double buffered channel



### Single buffered channel

In single buffer configuration the data at the memory side is accessed in a linear matter starting from the base address IDMABASE0. When the IDMA has finished transferring all data the and the DPSM has completed the transfer the DATAEND flag is set.

### Double buffered channel

In double buffer configuration the data at the memory side is subsequently accessed from 2 buffers, one located from base address IDMABASE0 and a second located from base address IDMABASE1. This allows firmware to process one memory buffer while the IDMA is accessing the other memory buffer. The size of the memory buffers is defined by IDMABSIZE. The buffer size shall be an integer multiple of the burst size. It is possible to update the base address of the buffers on-the-fly when the channel is enabled, the following rule apply:

- When IDMABACT bit is '0' the IDMA hardware uses the IDMABASE0 to access memory. When attempting to write to this register by Firmware the write is discarded, IDMABASE0 data will not be changed. Firmware is allowed to write IDMABASE1.
- When IDMABACT bit is '1' the IDMA hardware uses the IDMABASE1 to access memory. When attempting to write to this register by Firmware the write is discarded, IDMABASE1 data will not be changed. Firmware is allowed to write IDMABASE0.

When the IDMA has finished transferring the data of one buffer the buffer transfer complete flag (IDMABTC) is set and the IDMABACT bit toggles where after the IDMA continues transferring data from the other buffer. When the IDMA has finished transferring all data and the DPSM has completed the transfer the DATAEND flag is set.

The IDMABASE<sub>n</sub> address shall be word aligned.

### IDMA transfer error management

An IDMA transfer error can occur:

- When reading or writing a reserved address space.

On a IDMA transfer error subsequent IDMA transfers are disabled and an IDMATE flag is set. Depending when the IDMA transfer error occurs, it will normally cause the generation of a TXUNDERR or RXOVERR error.

The behavior of the IDMATE flag depend on when the IDMA transfer error occurs during the SDMMC transfer:

- An IDMA transfer error is detected before any SDMMC transfer error (TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT):
  - The IDMATE flag is set at the same time as the SDMMC transfer error flag.
  - The TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT interrupt is generated.
- An IDMA transfer error is detected during a STOP\_TRANSMISSION command:
  - The IDMATE flag is set at the same time as the DABORT flag.
  - The DABORT interrupt is generated.
- An IDMA transfer error is detected at the end of the SDMMC transfer (HOLD, or DATAEND).
  - The IDMATE flag is set at the end of the SDMMC transfer.
  - A SDMMC transfer end interrupt is generated and a HOLD or DATAEND flag is set.

The IDMATE will be generated on an other SDMMC transfer interrupt (TXUNDERR, RXOVERR, DCRCFAIL, DTIMEOUT, DABORT, HOLD, or DATAEND).

#### 48.4.7 AHB and SDMMC\_CK clock relation

The AHB shall at least have 3x more bandwidth than the SDMMC bus bandwidth i.e. for SDR50 4-bit mode (50Mbyte/s) the minimum sdmmc\_hclk frequency is 37.5MHz (150Mbyte/s).

**Table 308. AHB and SDMMC\_CK clock frequency relation**

SDMMC bus mode	SDMMC bus width	Maximum SDMMC_CK [MHz]	Minimum AHB clock [MHz]
eMMC DS	8	26	19.5
eMMC HS	8	52	39
eMMC DDR52	8	52	78
SD DS / SDR12	4	25	9.4
SD HS / SDR25	4	50	18.8
SD DDR50	4	50	37.5
SD SDR50	4	100	37.5

## 48.5 Card functional description

### 48.5.1 SD I/O mode

The following features are SDMMC specific operations:

- SDIO interrupts
- SDIO suspend/resume operation (write and read suspend)
- SDIO read wait operation by stopping the clock
- SDIO read wait operation by SDMMC\_D2 signaling

**Table 309. SDIO special operation control**

Operation mode	SDIOEN	RWMOD	RWSTOP	RWSTART	DTDIR
Interrupt detection	1	X	X	X	X
Suspend/Resume operation	X	X	X	X	X
ReadWait SDMMC_CK clock stop (START)	X	1	0	1	1
ReadWait SDMMC_CK clock stop (STOP)	X	1	1	1	1
ReadWait SDMMC_D2 signaling (START)	X	0	0	1	1
ReadWait SDMMC_D2 signaling (STOP)	X	0	1	1	1

### SD I/O interrupts

To allow the SD I/O card to interrupt the host, an interrupt function is available on pin 8 (shared with SDMMC\_D1 in 4-bit mode) on the SD interface. The use of the interrupt is optional for each card or function within a card. The SD I/O interrupt is level-sensitive, which means that the interrupt line must be held active (low) until it is either recognized and acted upon by the host or de-asserted due to the end of the interrupt period. After the host has serviced the interrupt, the interrupt status bit is cleared via an I/O write to the appropriate bit in the SD I/O card internal registers. The interrupt output of all SD I/O cards is active low and the application must provide external pull-up resistors on all data lines (SDMMC\_D[3:0]).

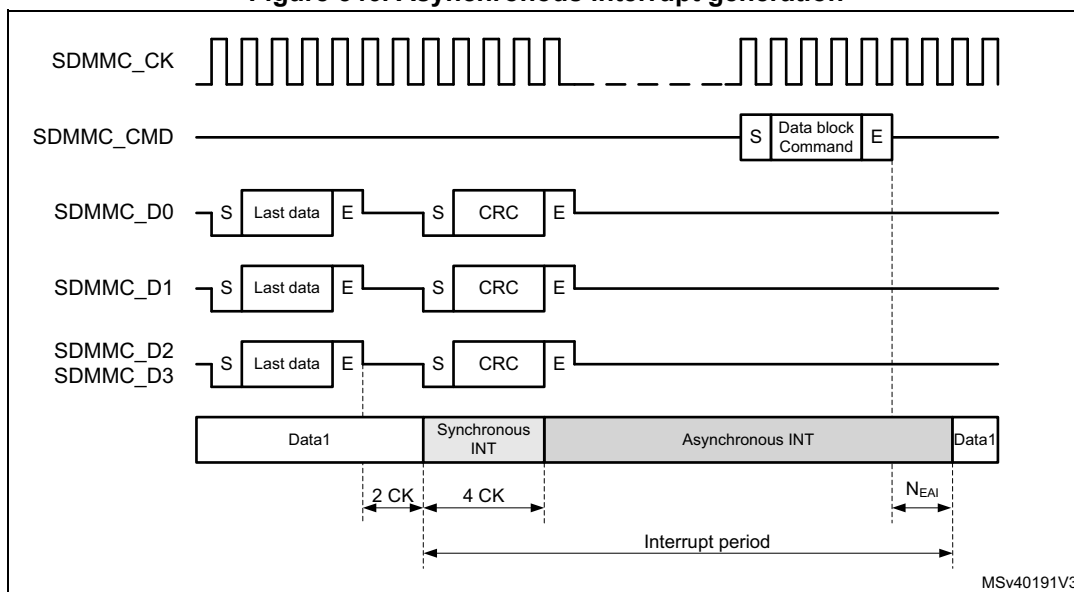
In SD 1-bit mode pin 8 is dedicated to the interrupt function (IRQ), and there are no timing constraints on interrupts.

In SD 4-bit mode the host samples the level of pin 8 (SDMMC\_D1/IRQ) into the interrupt detector only during the interrupt period. At all other times, the host interrupt ignores this value. The interrupt period begins when interrupts are enabled at the card and SDIOEN bit is set see register settings in [Table 309](#).

In 4-bit mode the card can generate a synchronous or asynchronous interrupt as indicated by the card CCCR register SAI and EAI bits.

- Synchronous interrupt, require the SDMMC\_CK to be active.
- Asynchronous interrupt, can be generated when the SDMMC\_CK is stopped, 4 cycles after the start of the card interrupt period following the last data block.

**Figure 546. Asynchronous interrupt generation**



The timing of the interrupt period is depended on the bus speed mode:

In DS, HS, SDR12, and SDR25 mode, selected by register bit BUSSPEED, the interrupt period is synchronous to the SD clock.

- The interrupt period ends at the next clock from the End bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set.
- The interrupt period resumes 2 SDMMC\_CK after the completion of the data block.
- At the data block gap the interrupt period is limited to 2 SDMMC\_CK cycles.

Note: DTEN shall not be used to start data transfer with SD and eMMC cards.

Figure 547. Synchronous interrupt period data read

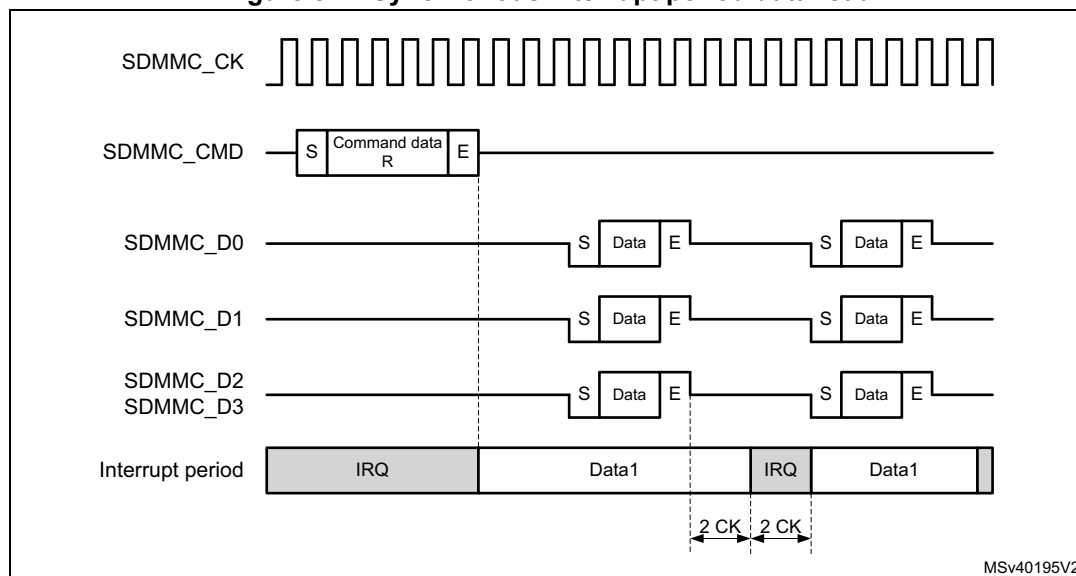
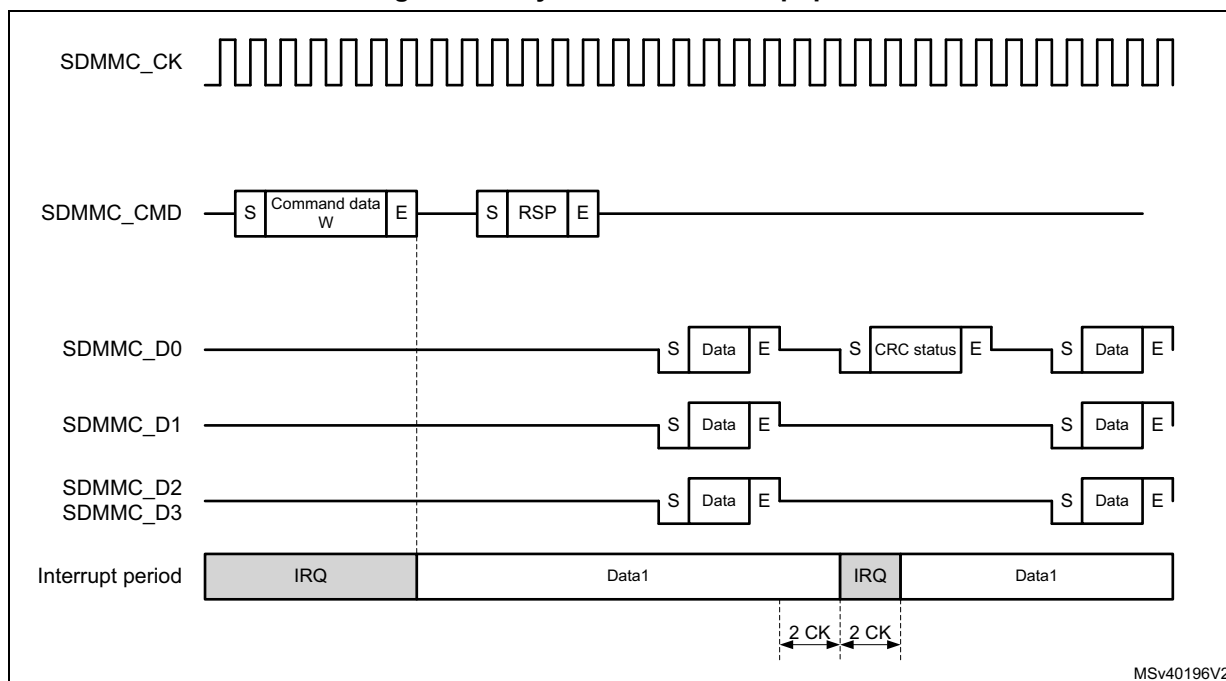


Figure 548. Synchronous interrupt period data write



In SDR50 and DDR50, selected by register bit BUSSPEED, due to propagation delay from the card to host, the interrupt period is asynchronous.

- The card interrupt period ends after 0 to 2 SDMMC\_CK cycles after the End bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set. At the host the interrupt period ends after the End bit of a command that transfers data block(s). A card interrupt issued in the 1 to 2 cycles after the command End bit are not detected by the host during this interrupt period.
- The card interrupt period resumes 2 to 4 SDMMC\_CK after the completion of the last data block. The host will resume the interrupt period always 2 cycles after the last data block.
- There is NO interrupt period at the data block gap.

Note: DTEN shall not be used to start data transfer with SD and eMMC cards.

Figure 549. Asynchronous interrupt period data read

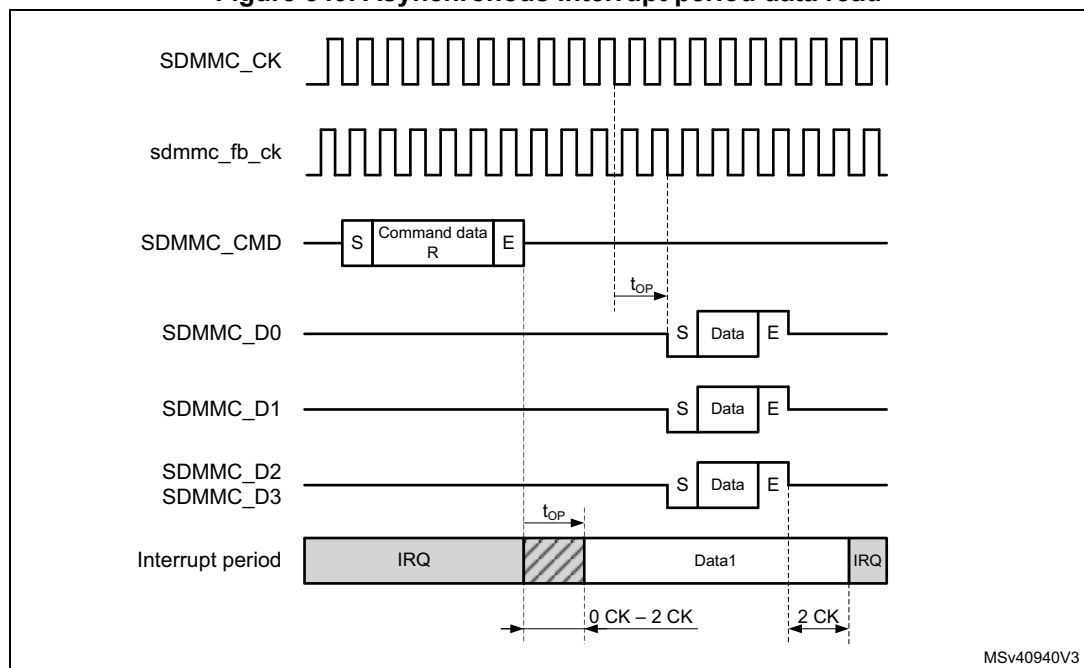
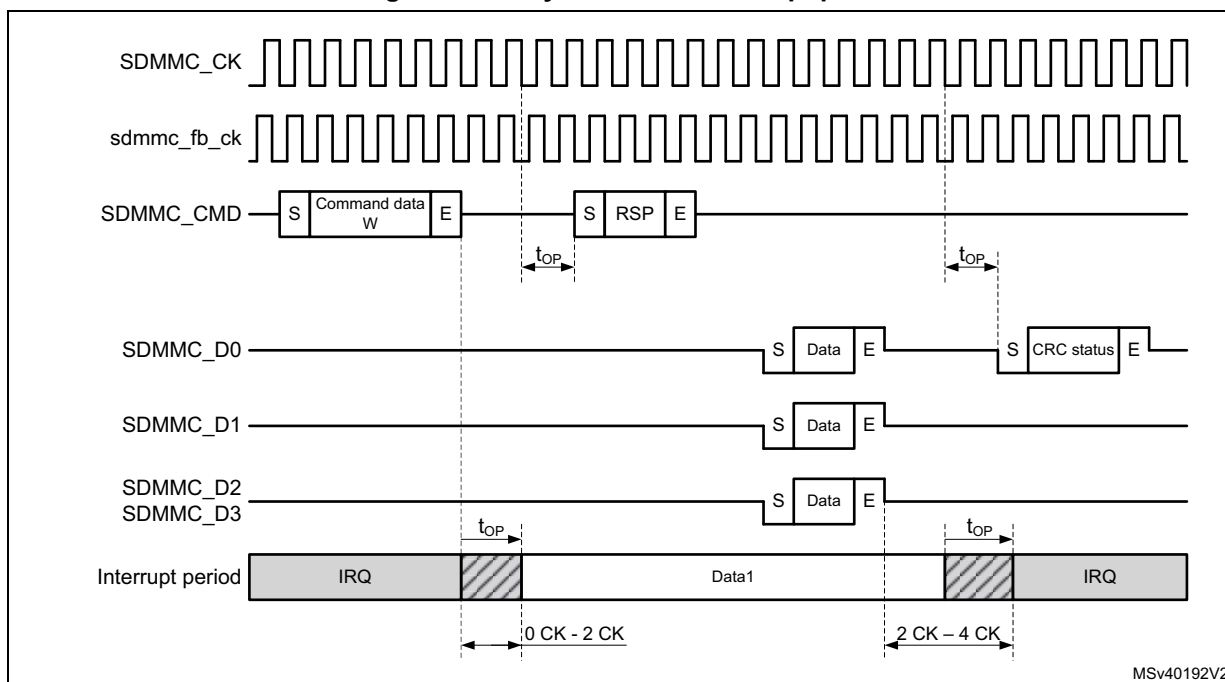


Figure 550. Asynchronous interrupt period data write



When transferring Open-ended multiple block data and using DTMODE “block data transfer ending with STOP\_TRANSMISSION command”, the SDMMC will mask the interrupt period after the last data block until the end of the CMD12 STOP\_TRANSMISSION command.

The interrupt period is applicable for both memory and I/O operations.

In 4-bit mode interrupts can be differentiated from other signaling according [Table 310](#).

Table 310. 4-bit mode Start, interrupt, and CRC-status Signaling detection

SDMMC data line	Start	Interrupt	CRC-status
SDMMC_D0	0	1 or CRC-status	0
SDMMC_D1	0	0	X
SDMMC_D2	0	1 or ReadWait	X
SDMMC_D3	0	1	X

### SD I/O suspend and resume

This function is NOT supported in SDIO version 4.00 or later.

Within a multifunction SD I/O or a card with both I/O and memory functions, there are multiple devices (I/O and memory) that share access to the eMMC/SD bus. To share access to the host among multiple devices, SD I/O and combo cards optionally implement the concept of suspend/resume. When a card supports suspend/resume, the host can temporarily halt (suspend) a data transfer operation to one function or memory to free the bus for a higher-priority transfer to a different function or memory. After this higher-priority transfer is complete, the original transfer is restarted (resume) where it left off.

To perform the suspend/resume operation on the bus, the host performs the following steps:

1. Determines the function currently using the SDMMC\_D[3:0] line(s)
2. Requests the lower-priority or slower transaction to suspend
3. Waits for the transaction suspension to complete
4. Begins the higher-priority transaction
5. Waits for the completion of the higher priority transaction
6. Restores the suspended transaction

The card receiving a suspend command will respond with its current bus status. Only when the bus has been suspended by the card the bus status will indicate suspension completed.

There are different suspend cases conditions:

- Suspend request accepted prior to the start of data transfer.
- Suspend request not accepted, (due to data being transferred at the same time), the host keeps checking the request until it is accepted. (data transfer has suspended)
- Suspend request during write busy.
- Suspend request with write multiple.
- Suspend request during ReadWait.

For the host to know if the bus has been released it shall check the status of the suspend request, suspension completed.

When the bus status of the suspend request response indicates suspension completed, the card has released the bus. At this time the state of the suspended operation shall be saved where after another operation can start.

The suspend command shall be sent with the CMDSPEND bit set. This allows to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

The hardware does not save the number of remaining data to be transferred when resuming the suspended operation. It is up to firmware to determine the data that has been transferred and resume with the correct remaining number of data bytes.

While receiving data from the card, the SDMMC can suspend the read operation after the read data block end (DPSM in Wait\_R). After receiving the suspend acknowledgment response from the card the following steps shall be taken by firmware:

1. The normal receive process shall be stopped by setting DTHOLD bit.
  - a) The remaining number of data bytes in the FIFO shall be read until the receive FIFO is empty (RXFIFOE flag is set), and when IDMAEN = 0 the FIFO shall be reset with FIFORST.
2. The confirmation that all data has been read from the FIFO, and that the suspend is completed is indicated by the DHOLD flag.
  - a) The remaining number of data bytes (multiple of data blocks) still to be read when resuming the operation shall be determined from the remaining number of bytes indicated by the DATACOUNT.

*Note:* When a DTIMEOUT flag occurs during the suspend procedure, this shall be ignored.

To resume receiving data from the card, the following steps shall be taken by firmware:

1. The remaining number of data bytes (multiple of data blocks) shall be programmed in DATALENGTH.
2. The DPSM shall be configured to receive data in the DTDIR bit.
3. The resume command shall be sent from the CPSM, with the CMDTRANS bit set and the CMDSUSPEND bit set, which will end the interrupt period when data transfer is resumed (response bit DF = 1) and enabled the DPSM, after which the card will resume sending data.

While sending data to the card, the SDMMC can suspend the write operation after the write data block CRC status end (DPSM in Busy). Before sending the suspend command to the card the following steps shall be taken by firmware:

1. Enable DHOLD flag (and DBCKEND flag when IDMAEN = 0)
2. The DPSM shall be prevented from start sending a new data block by setting DTHOLD.
3. When IDMAEN = 0: When receiving the DBCKEND flag the data transfer is stopped. Firmware can stop filling the FIFO, after which the FIFO shall be reset with FIFORST. Any bytes still in the FIFO need to be rewritten when resuming the operation.
4. When receiving the DHOLD flag the data transfer is stopped. The remaining number of data bytes still to be written when resuming shall be determined from the remaining number of bytes indicated by the DATACOUNT.
5. To suspend the card the suspend command shall be sent by the CPSM with the CMDSUSPEND bit set. This allows to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

To resume sending data to the card, the following steps shall be taken by firmware:

1. The remaining number of data bytes shall be programmed in DATALENGTH.
2. The DPSM shall be configured for transmission with DTDIR set and enabled by having the CPSM send the resume command with the CMDTRANS bit set and the CMDSUSPEND bit set. This will end the interrupt period and start the data transfer. The DPSM will either go to the Wait\_S state when SDMMC\_D0 does not signal busy, or will go to the Busy state when busy is signaled.
3. When IDMAEN = 1: The IDMA needs to be reprogrammed for the remaining bytes to be transferred.
4. When IDMAEN = 0: Firmware shall start filling the FIFO with the remaining data.

#### SD I/O ReadWait

There are 2 methods to pause the data transfer during the Block gap:

1. Stopping the SDMMC\_CK.
2. Using ReadWait signaling on SDMMC\_D2.

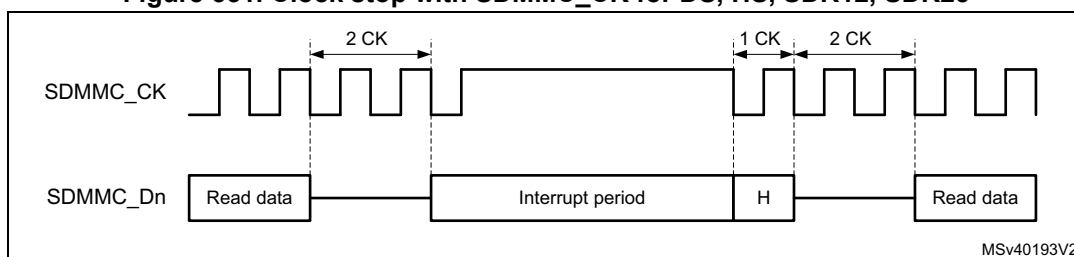
The SDMMC can perform a ReadWait with register settings according [Table 309](#).

Depending the SDMMC operation mode (DS, HS, SDR12, SDR25) or (SDR50, DDR) each method has a different characteristic.

The timing for pause read operation by stopping the SDMMC\_CK for DS, HS, SDR12, and SDR25, the SDMMC\_CK may be stopped 2 SDMMC\_CK cycles after the End bit. When ready the host resumes by restarting clock, see [Figure 551](#).

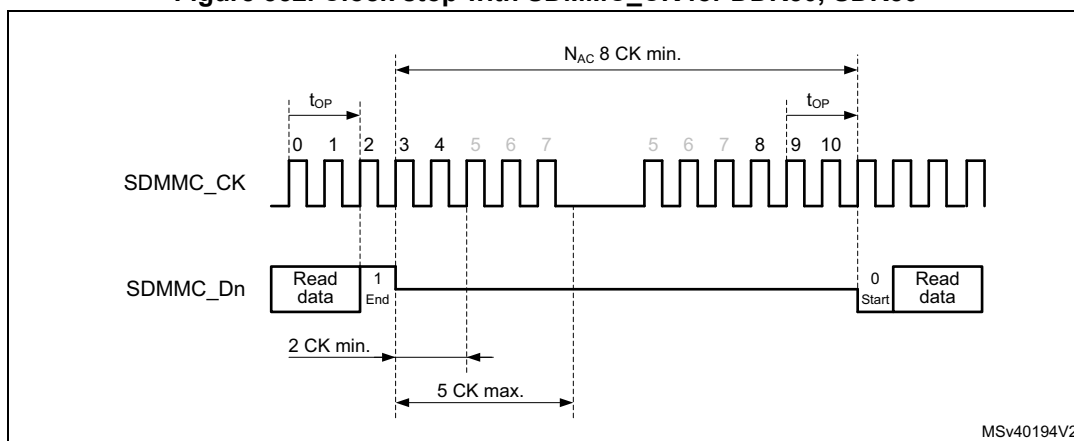


Figure 551. Clock stop with SDMMC\_CK for DS, HS, SDR12, SDR25



The timing for pause read operation by stopping the SDMMC\_CK for SDR50 and DDR50, the SDMMC\_CK may be stopped minimum 2 SDMMC\_CK cycles and maximum 5 SDMMC\_CK cycles, after the End bit. When ready the host resumes by restarting clock, see [Figure 552](#). (In DDR50 mode the SDMMC\_CK shall only be stopped after the falling edge, when the clock line is low.)

Figure 552. Clock stop with SDMMC\_CK for DDR50, SDR50



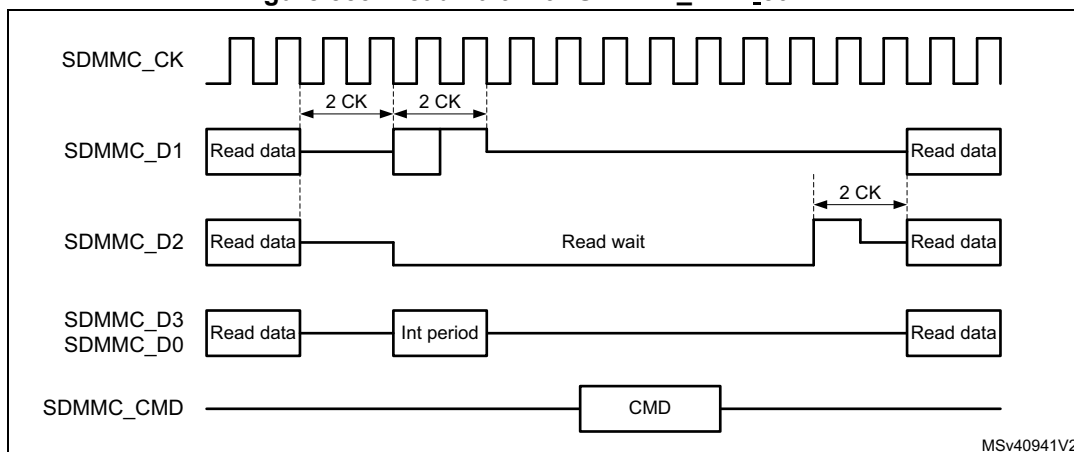
In ReadWait SDMMC\_CK clock stopping, when RWSTART is set, the DSPM stops the clock after the End bit of the current received data block CRC. The clock start again after writing 1 to the RWSTOP bit, where after the DPSM waits for a Start bit from the card.

As SDMMC\_CK is stopped, no command can be issued to the card. During a ReadWait interval, the SDMMC can still detect SDIO interrupts on SDMMC\_D1.

The optional ReadWait signaling on SDMMC\_D2 (RW) operation is defined only for the SD 1-bit and 4-bit modes. The ReadWait operation allows the host to signal a card that is reading multiple registers (IO\_RW\_EXTENDED, CMD53) to temporarily stall the data transfer while allowing the host to send commands to any function within the SD I/O device. To determine when a card supports the ReadWait protocol, the host must test capability bits in the internal card registers.

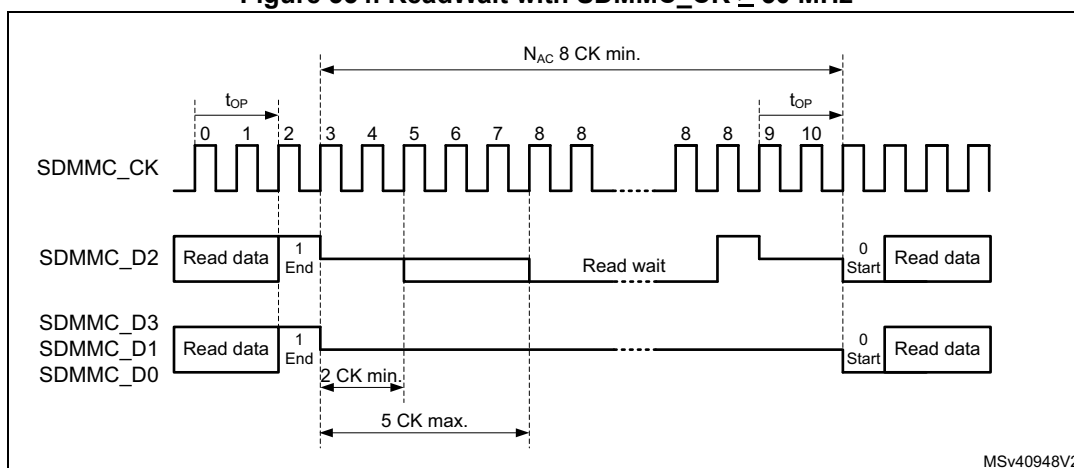
The timing for ReadWait with a SDMMC\_CK less then 50MHz (DS, HS, SDR12, SDR25) is based on the interrupt period generated by the card on SDMMC\_D1. The host by asserting SDMMC\_D2 low during the interrupt period requests the card to enter ReadWait. To exit ReadWait the host shall raise SDMMC\_D2 high during one SDMMC\_CK cycles before making it Hi-Z, see [Figure 553](#).

Figure 553. ReadWait with SDMMC\_CK < 50 MHz



For SDR50 with a SDMMC\_CK more than 50MHz, and DDR50, the card will treat the ReadWait request on SDMMC\_D2 as an asynchronous event. The host by asserting SDMMC\_D2 low after minimum 2 SDMMC\_CK cycles and maximum 5 SDMMC\_CK cycles, request the card to enter ReadWait. To exit ReadWait the host shall raise SDMMC\_D2 high during one SDMMC\_CK cycles before making it Hi-Z. The host shall raise SDMMC\_D2 on the SDMMC\_CK clock (see Figure 554).

Figure 554. ReadWait with SDMMC\_CK ≥ 50 MHz



In ReadWait SDMMC\_D2 signaling, when RWSTART is set, the DPSM drives SDMMC\_D2 after the End bit of the current received data block CRC. The ReadWait signaling on SDMMC\_D2 will be removed when writing 1 to the RWSTOP bit. The DPSM remains in ReadWait state for two more SDMMC\_CK clock cycles to drive SDMMC\_D2 to 1 for one clock cycle (in accordance with SDIO specification), where after the DPSM waits for a Start bit from the card.

During the ReadWait signaling on SDMMC\_D2 commands can be issued to the card. During the ReadWait interval, the SDMMC can detect SDIO interrupts on SDMMC\_D1.

### 48.5.2 CMD12 send timing

CMD12 is used to stop/abort the data transfer, the card data transmission is terminated two clock cycles after the End bit of the Stop Transmission command.

Table 311. CMD12 use cases

Data operation	Stop Transmission command CMD12 Description
SDMMC stream write	The data transfer is stopped/aborted by sending the Stop Transmission command.
SDMMC open ended multiple block write	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block write with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block write. (sending the Stop Transmission command after the card has received the last block is regarded as an illegal command.) If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC stream read	The data transfer is stopped/aborted by sending the Stop Transmission command.
SDMMC open ended multiple block read	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block read with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block read. (sending the Stop Transmission command after the card has transmitted the last block is regarded as an illegal command.) Transaction can be aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.

All data write and read commands can be aborted any time by a Stop Transmission command CMD12. The following data abort procedure applies during an ongoing data transfer:

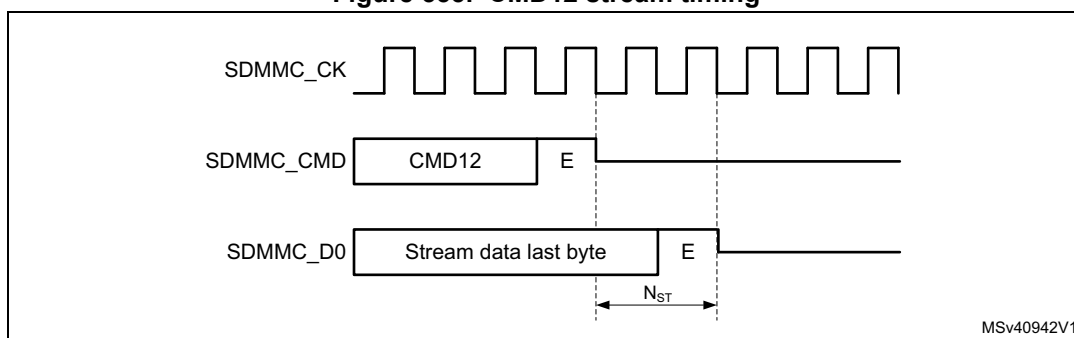
1. Load CMD12 Stop Transmission command in registers and set the CMDSTOP bit.
  - a) This causes the CPSM to generate the Abort signal when the command is sent to the DPSM.
2. Configure the CPSM to send a command immediately (clear WAITPEND bit).
  - a) The card, when sending data, will stop data transfer 2 cycles after the Stop Transmission command End bit.  
The card when no data is being sent, will not start sending any new data.
  - b) The host, when sending data, will send one last data bit followed by an End bit after the Stop Transmission command End bit.  
The host when not sending data, will not start sending any new data.
3. When IDMAEN = 0, the FIFO need to be reset with FIFORST.
  - a) When writing data to the card. On the CMDREND flag FW shall stop writing data to the FIFO. Subsequently the FIFO shall be reset with FIFORST, this will flush the FIFO.
  - b) When reading data from the card. On the CMDREND flag FW shall read the remaining data from the FIFO. Subsequently the FIFO shall be reset with FIFORST.
4. When IDMAEN = 1, hardware will take care of the FIFO.
  - a) When writing data to the card. On the Abort signal hardware will stop the IDMA and subsequently the FIFO will be flushed.
  - b) When reading data from the card. On the Abort signal hardware will instruct the IDMA to transfer the remaining data from the FIFO to RAM.
5. When the FIFO is empty/reset the DABORT flag will be generated.

### Stream operation and CMD12

To stop the stream transfer after the last byte to be transferred, the CMD12 End bit timing shall be sent aligned with the data stream end of last byte. The following write stream data procedure applies:

1. Initialize the stream data in the DPSM, DTMODE = MCC stream data transfer.
2. Send the WRITE\_DATA\_STREAM command from the CPSM with CMDTRANS = 1.
3. Preload CMD12 in command registers, with the CMDSTOP bit shall set.
4. Configure the CPSM to send a command only after a wait pending (WAITPEND = 1) end of last data (according DATALENGTH).
5. Enabling the CPSM to send the STOP\_TRANSMISSION command, the stream data End bit and command End bit will be aligned.
  - a) When DATALENGTH > 5 bytes, Command CMD12 will be waited in the CPSM to be aligned with the data transfer End bit.
  - b) When DATALENGTH < 5 bytes, Command CMD12 will be started before and the DPSM will remain in the Wait\_S state to align the data transfer end with the CMD12 End bit.
6. The write stream data can be aborted any time by clearing the WAITPEND bit. This will cause the Preloaded CMD12 to be sent immediately and stop the write data stream.

Figure 555. CMD12 stream timing



To stop the read stream transfer after the last byte, the CMD12 End bit timing shall occur after the last data stream byte. The following read stream data procedure applies:

1. Wait for all data to be received by the DPSM (DATAEND flag).
  - a) The DPSM will not receive more data than indicated by DATALENGTH, even if the card is sending more data.
2. Send CMD12 by the CPSM.
  - a) CMD12 will stop the card sending data.

*Note:* The SDMMC will not receive any more data from the card when  $DATACOUNT = 0$ , even when the card continues sending data.

### Block operation and CMD12

To stop block transfer at the end of the data, the CMD12 End bit shall be sent after the last block End bit.

When writing data to the card the CMD12 End bit shall be sent after the write data block CRC token End bit. This requires the CMD12 sending to be tied to the data block transmission timing. To stop an Open-ended Multiple block write, the following procedure applies:

1. Before starting the data transfer, set DTMODE to "block data transfer ending with STOP\_TRANSMISSION command".
2. Wait for all data to be sent by the DPSM and the CRC token to be received, (DATAEND flag).
  - a) The DPSM will not send more data than indicated by DATALENGTH.
3. Send CMD12 by the CPSM.
  - a) CMD12 will set the card to Idle mode.

When reading data from the card the CMD12 End bit shall be sent earliest at the same time as the card read data block last data bit. This requires the CMD12 sending to be tied to the data block reception timing. The following stop Open-ended Multiple block read data block procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP\_TRANSMISSION command”.
2. Wait for all data to be received by the DPSM (DATAEND flag).
  - a) The DPSM will not receive more data than indicated by DATALENGTH, even if the card is sending more data.
3. Send CMD12 with CMDSTOP bit set by the CPSM.
  - a) CMD12 will stop the Card sending more data and set the card to Idle mode. Any ongoing block transfer will be aborted by the Card.

*Note:* The SDMMC will not receive any more data from the card when `DATACOUNT = 0`, even when the card continues sending data.

### 48.5.3 Sleep (CMD5)

The eMMC card may be switched between a Sleep state and a Standby state by CMD5. In the Sleep state the power consumption of the card is minimized and the Vcc power supply may be switched off.

The CMD5 (SLEEP) is used to initiate the state transition from Standby state to Sleep state. The card indicates Busy, pulling down SDMMC\_D0, during the transition phase. The Sleep state is reached when the card stops pulling down the SDMMC\_DO line.

To set the card into Sleep state the following procedure applies:

1. Enable interrupt on BUSYD0END.
2. Send CMD5 (SLEEP).
3. On BUSYD0END interrupt, card is in Sleep state
4. Vcc power supply is allowed to be switched off

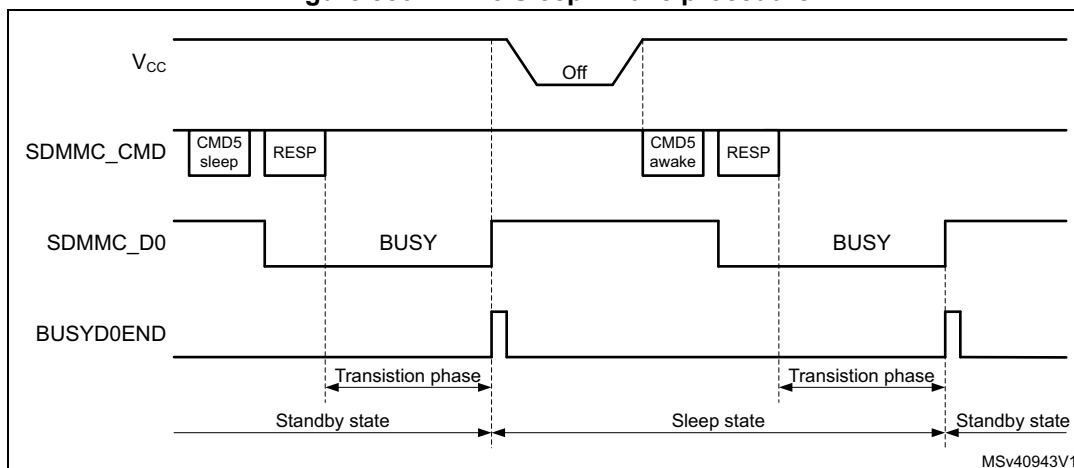
The CMD5 (AWAKE) is used to initiate the state transition from Sleep state to Standby state. The card indicates Busy, pulling down SDMMC\_D0, during the transition phase. The Standby state is reached when the card stops pulling down the SDMMC\_DO line.

To set the card into Sleep state the following procedure applies:

1. Switch on Vcc power supply and wait until minimum operating level is reached.
2. Enable interrupt on BUSYD0END.
3. Send CMD5 (AWAKE).
4. On BUSYD0END interrupt card is in Standby state.

The Vcc power supply is allowed to be switched off only after the Sleep state has been reached. The Vcc supply shall be reinstalled before CMD5 (AWAKE) is sent.

Figure 556. CMD5 Sleep Awake procedure



### 48.5.4 Interrupt mode (Wait-IRQ)

The host and card enter and exit interrupt mode (Wait-IRQ) simultaneously. In interrupt mode there is no data transfer. The only message allowed is an interrupt service request response from the card or the host. For the interrupt mode to work correctly the SDMMC\_CLK frequency shall be set in accordance with the achievable SDMMC\_CMD data rate in Open Drain mode, which depend on the capacitive load and pull-up resistor. The CLKDIV shall be set >1, and the SETCLKRX shall select either the sdmmc\_io\_in\_ck or SDMMC\_CLKin source.

The host must ensure that the card is in Standby state before issuing the CMD40 (GO\_IRQ\_STATE). While waiting for an interrupt response the SDMMC\_CLK clock signal must be kept active.

A card in interrupt mode (IRQ state):

- is waiting for an internal card interrupt event. Once the event occurs, the card starts to send the interrupt service request response. The response is sent in open-drain mode.
- while waiting for the internal card interrupt event, the card also monitors the SDMMC\_CMD line for a Start bit. Upon detection of a Start bit the card will abort the interrupt mode and switch to Standby state.

The host in interrupt mode (CPSM Wait state waiting for interrupt):

- is waiting for a card interrupt service request response (Start bit).
- while waiting for a card interrupt service request response the host may abort the interrupt mode (by clearing the WAITINT register bit), which causes the host to send a interrupt service request response R5 with RCA = 0x0000 in open-drain mode.

When sending the interrupt service request response, the sender bit-wise monitors the SDMMC\_CMD bit stream. The sender whose interrupt service request response bit does not correspond to the bit on the SDMMC\_CMD line stops sending. In the case of multiple senders only one will successfully send its full interrupt service request response. i.e. If the host sends simultaneously, it will lose sending after the transmission bit.

To handle the interrupt mode, the following procedure applies:

1. Set the SDMMC\_CK frequency in accordance with the achievable SDMMC\_CMD data rate in Open-drain mode, CLKDIV shall be set >1, and SETCLKRX shall select the sdmmc\_io\_in\_ck.
2. Load CMD40 (GO\_IRQ\_STATE) in the command registers.
3. Enable wait for interrupt by setting WAITINT register bit.
4. Configure the CPSM to send a command immediately.
  - a) This will cause the CMD40 to be sent and the CPSM to be halted in the Wait state, waiting for a interrupt service request response.
5. To exit the wait for interrupt state (CPSM Wait state):
  - a) Upon the detection of an interrupt service request response Start bit the CPSM moves to the Receive state where the response is received. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.
  - b) To abort the interrupt mode the host clears the WAITINT register bit, which will cause the host to send an interrupt service request response by its self. Which will move the CPSM to the Receive state. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.

*Note:* On a simultaneous send interrupt service request response Start bit collision the host will lose the bus access after the Transmission bit.

### 48.5.5 Boot operation

In boot operation mode the host can read boot data from the card by either one of the 2 boot operation functions:

1. Normal boot. (keeping CMD line low)
2. Alternative boot (sending CMD0 with argument 0xFFFFFFFF)

The boot data can be read according the following configuration options, depending on card register settings:

- The partition from which boot data is read (EXT\_CSD Byte[179])
- The boot data size (EXT\_CSD Byte[226])
- The bus configuration during boot (EXT\_CSD Byte[177])
- Receiving boot acknowledgment from the card. (EXT\_CSD Byte[179])

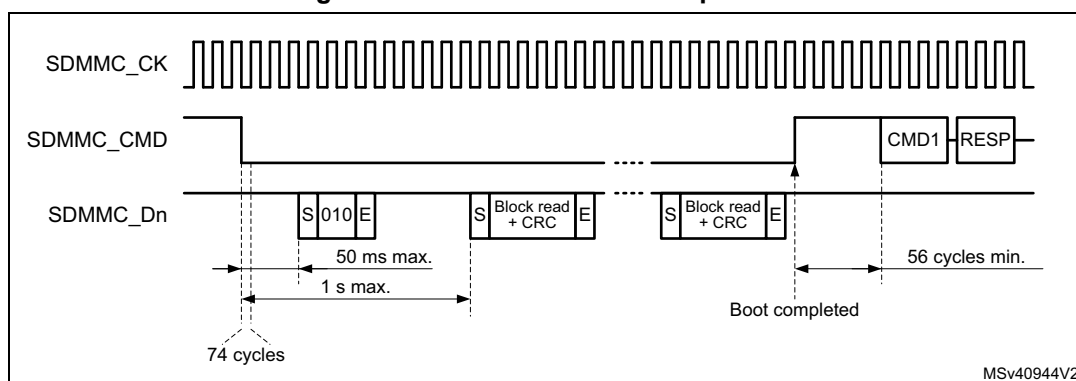
If boot acknowledgment is enabled the card send pattern 010 on SDMMC\_D0 within 50ms after boot mode has been requested by either CMD line going low or after CMD0 with argument 0xFFFFFFFF. A boot acknowledgment timeout (ACKTIMEOUT) and acknowledgment status (ACKFAIL) is provided.

#### Normal boot operation

If the SDMMC\_CMD line is held low for at least 74 clock cycles after card power-up or reset, before the first command is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD line goes low, the card starts to sent the first boot code data on the SDMMC\_Dn line(s). The host must keep the SDMMC\_CMD line low until after all boot data has been read. The host can terminate boot mode by pulling the SDMMC\_CMD line high.



Figure 557. Normal boot mode operation



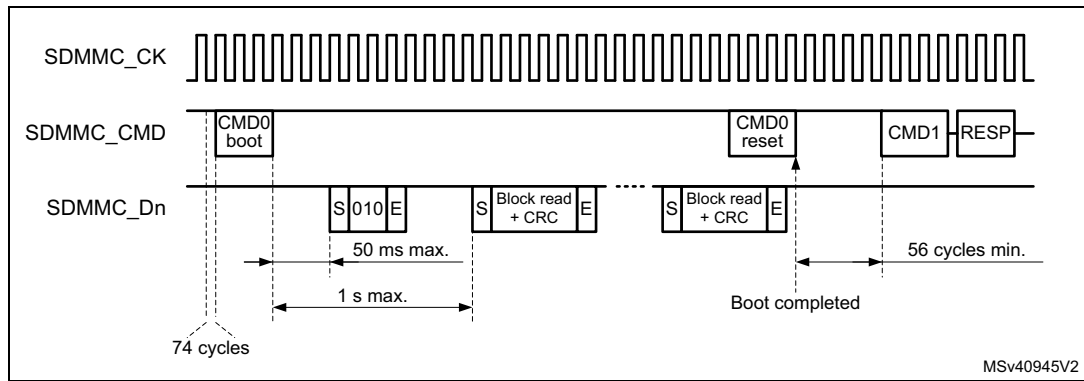
To perform the normal boot procedure the following steps needed:

1. Reset the card.
2. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKFAIL and ACKTIMEOUT interrupt.
3. enable the data reception by setting the DPSM in receive mode (DTPDIR) and the number of data bytes to be received in DATALENGTH.
4. Enable the DTIMEOUT, DATAEND, and CMDSENT interrupts for end of boot command confirmation.
5. Select the normal boot operation mode in BOOTMODE, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN. This will cause:
  - the SDMMC\_CMD to be driven low. (BOOTMODE = normal boot).
  - the ACK timeout to start.
  - DPSM to be enabled.
6. The incorrect reception of the boot acknowledgment can be detected with ACKFAIL flag or ACKTIMEOUT flag when enabled.
  - when an incorrect boot acknowledgment is received the ACKFAIL flag occurs.
  - when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
7. when all boot data has been received the DATAEND flag will occur.
  - when data CRC fails the DCRCFAIL flag is also generated.
  - when the data timeout occurs the DTIMEOUT flag is also generated.
8. When last data has been received, read data from the FIFO until FIFO is empty (RXFIFOE = 1) after which end of data DATAEND flag is generated.
  - SDMMC has completely received all data and the DPSM is disabled.
9. The boot procedure will be terminated by FW clearing BOOTEN, which will cause the SDMMC\_CMD line to go high. The CMDSENT flag is generated 56 cycles later to indicate that a new command can be sent.
  - a) If the boot procedure is aborted by FW before all data has been received the CPSM Abort signal will stop data reception and disable the DPSM which will generate an DABORT flag when enabled.
10. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command.

**Alternative boot operation**

After card power-up or reset, if the host send CMD0 with the argument 0xFFFFFFFF after 74 clock cycles before CMD0 is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD0 with argument 0xFFFFFFFF has been sent, the card starts to send the first boot code data on the SDMMC\_Dn line(s). The master terminates boot operation by sending CMD0 (Reset).

**Figure 558. Alternative boot mode operation**



To perform the alternative boot procedure the following steps needed:

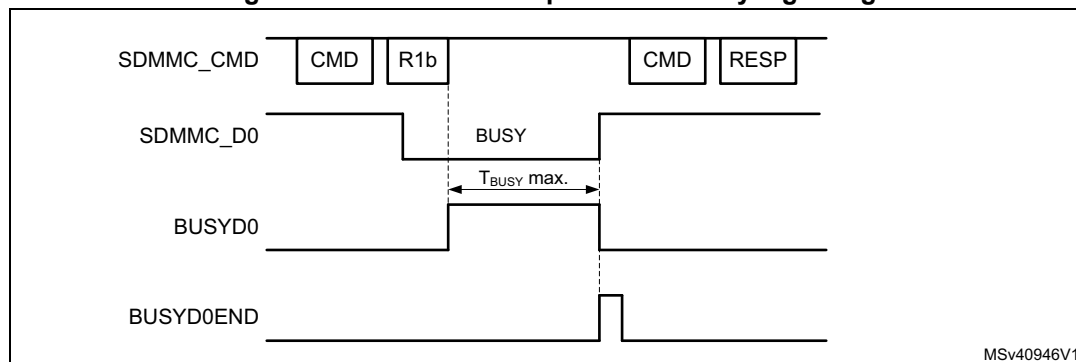
1. Move the SDMMC to power-off state, and reset the card
2. Move the SDMMC to power-on state. This will guarantee the 74 SCDMMC\_CK cycles to be clocked before any command.
3. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKTIMEOUT flag.
4. enable the data reception by setting the DPSM in receive mode (DTPDIR) and the number of data to be received in DATALENGTH. Enable the DTIMEOUT and DATAEND flags.
5. Select the alternative boot operation mode in BOOTMODE, load the CMD0 with the 0xFFFFFFFF argument in the command registers. Enable CMDSENT flag for end of

- boot command confirmation, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN. This will cause:
- the loaded command and argument to be sent out. (BOOTMODE = alternative boot).
  - the ACK timeout to start.
  - DPSM to be enabled.
6. When the command has been sent the CMDSENT flag is generated, at which time the BOOTEN bit shall be cleared.
  7. the reception of the boot acknowledgment can be detected with ACKFAIL flag when enabled.
    - when the boot acknowledgment is not received in time the ACKTIMEOUT flag will occur.
  8. when all boot data has been received the DATAEND flag will occur.
    - when data CRC fails the DCRCFAIL flag is also generated.
    - when the data timeout occurs the DTIMEOUT flag is also generated.
  9. When last data has been received, read data from the FIFO until FIFO is empty (RXFIFOE = 1) after which end of data DATAEND flag is generated.
    - SDMMC has completely received all data and the DPSM is disabled.
  10. The BOOTEN bit shall be cleared, before terminating the boot procedure by sending CMD0 (Reset) with BOOTMODE = alternative boot. This will cause the CMDSENT flag to occur 56 cycles after the Command.
    - if the boot procedure is aborted by FW before all data has been received the CPSM Abort signal will stop the data transfer and disable the DPSM which will generate an DABORT flag when enabled.
  11. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command. When the RESET command has been sent successfully, the BOOTMODE control bit has to be cleared to terminate the boot operation.

### 48.5.6 Response R1b handling

When sending commands which have a R1b response the busy signaling is reflected in the BUSYD0 register bit and the release of busy with the BUSYD0END flag. The SDMMC\_D0 line is sampled at the end of the R1b response and signaled in the BUSYD0 register bit. The BUSYD0 register bit is reset to not busy when the SDMMC\_D0 line release busy, at the same time the BUSYD0END flag is generated.

Figure 559. Command response R1b busy signaling



MSv40946V1

The expected maximum busy time shall be set in the DATATIME register before sending the command. When enabled, the DTIMEOUT flag will be set when after the R1b response busy stays active longer then the programmed time.

To detect the SDMMC\_D0 busy signaling when sending a Command with R1b response the following procedure applies:

- Enable CMDREND flag
- Send Command through CPSM.
- On the CMDREND flag check the BUSYD0 register bit.
  - If BUSYD0 signals not busy, signal busy release to the Firmware
  - If BUSYD0 signals busy, wait for BUSYD0END flag
- On BUSYD0END flag signal busy released to the firmware.
- On DTIMEOUT flag busy is active longer then programmed time.

### 48.5.7 Reset and card cycle power

#### Reset

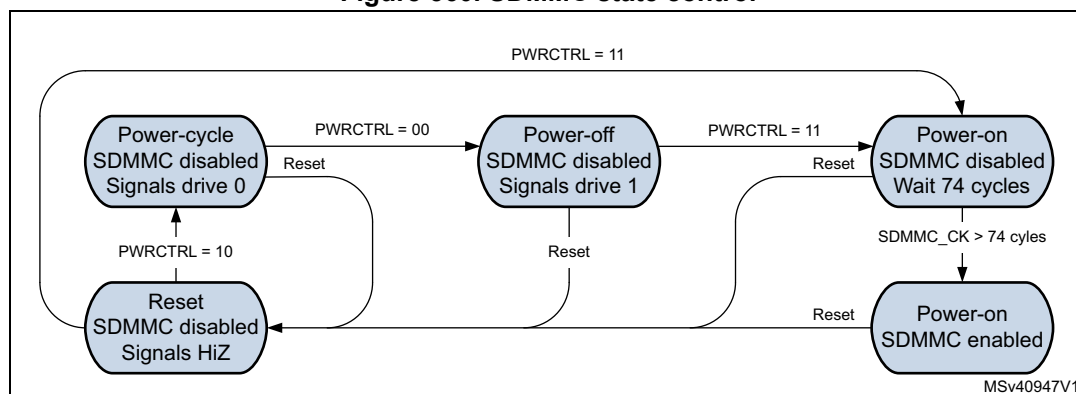
Following reset the SDMMC will be in the reset state. In this state the SDMMC is disabled and no command nor data can be transferred. The SDMMC\_D[7:0], and SDMMC\_CMD are in HiZ and the SDMMC\_CK is driven low.

Before moving to the power-on state the SDMMC shall be configured.

In the power-on state the SDMMC\_CK clock is running. First 74 SDMMC\_CK cycles will be clocked after which the SDMMC is enabled and command and data can be transferred.

The SDMMC states are controlled by Firmware with the PWRCTRL register bits according [Figure 560](#).

Figure 560. SDMMC state control

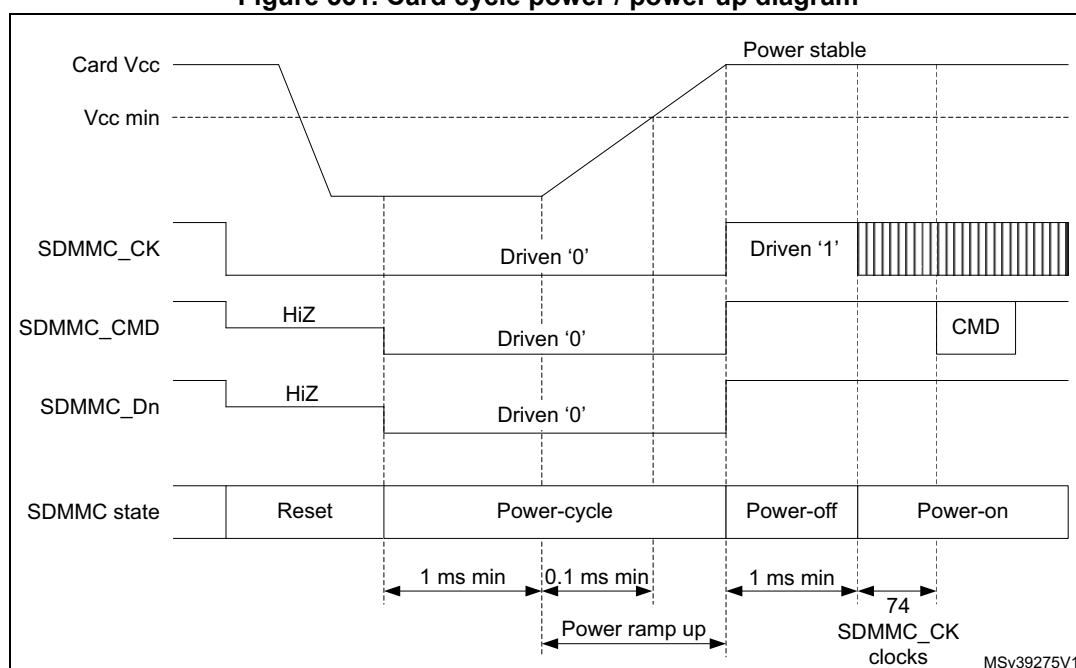


#### Card cycle power

To perform a card cycle power the following procedure applies:

1. Reset the SDMMC with the RCC.SDMMCxRST register bit. This will reset the SDMMC to the reset state and the CPSM and DPSM to the Idle state.
2. Disable the Vcc power to the card.
3. Set the SDMMC in power-cycle state. This will make that the SDMMC\_D[7:0], SDMMC\_CMD and SDMMC\_CK are driven low, to prevent the card from being supplied through the signal lines.
4. After minimum 1ms enable the Vcc power to the card.
5. After the power ramp period set the SDMMC to the power-off state for minimum 1ms. The SDMMC\_D[7:0], SDMMC\_CMD and SDMMC\_CK are set to drive "1".
6. After the 1ms delay set the SDMMC to power-on state in which the SDMMC\_CK clock will be enabled.
7. After 74 SDMMC\_CK cycles the first command can be sent to the card.

**Figure 561. Card cycle power / power up diagram**



## 48.6 Hardware flow control

The hardware flow control functionality is used to avoid FIFO underrun (TX mode) and overrun (RX mode) errors.

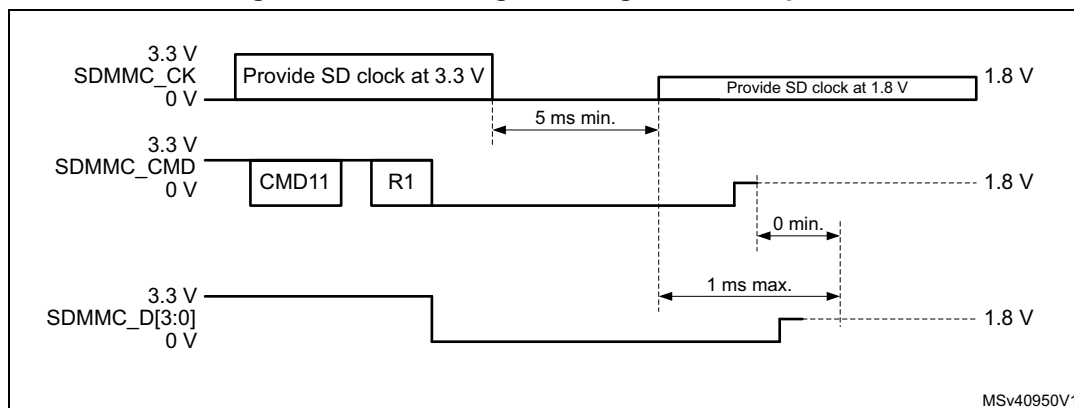
The behavior is to stop SDMMC\_CK during data transfer and freeze the SDMMC state machines. The data transfer is stalled when the FIFO is unable to transmit or receive data. The data transfer remains stalled until the transmit FIFO is half full or all data according DATALENGTH has been stored, or until the receive FIFO is half empty. Only state machines clocked by SDMMC\_CK are frozen, the AHB interfaces are still alive. The FIFO can thus be filled or emptied even if flow control is activated.

To enable hardware flow control, the HWFC\_EN register bit must be set to 1. After reset hardware flow control is disabled.

## 48.7 Ultra-high-speed phase I (UHS-I) voltage switch

UHS-I mode (SDR12, SDR25, SDR50, and DDR50) requires the support for 1.8V signaling. After power up the card starts in 3.3V mode. CMD11 invokes the voltage switch sequence to the 1.8V mode. When the voltage sequence is completed successfully the card enters UHS-I mode with default SDR12 and card input and output timings are changed.

**Figure 562. CMD11 signal voltage switch sequence**



To perform the signal voltage switch sequence the following steps are needed:

1. Before starting the Voltage Switch procedure, the SDMMC\_CK frequency shall be set in the range 100 kHz - 400 kHz.
2. The host starts the Voltage Switch procedure by setting the VSWITCHEN bit before sending the CMD11.
3. The card returns an R1 response.
  - if the response CRC is pass, the Voltage Switch procedure continues the host will no longer drive the CMD and SDMMC\_D[3:0] signals until completion of the voltage switch sequence. Some cycles after the response the SDMMC\_CK will be stopped and the CKSTOP flag will be set.
  - if the response CRC is fail (CCRCFAIL flag) or no response is received before the timeout (CTIMEOUT flag), the Voltage Switch procedure is stopped.
4. The card drives CMD and SDMMC\_D[3:0] to low at the next clock after the R1 response.
5. The host, after having received the R1 response, may monitor the SDMMC\_D0 line using the BUSYD0 register bit. The SDMMC\_D0 line is sampled two SDMMC\_CK clock cycles after the Response. The Firmware may read the BUSYD0 register bit following the CKSTOP flag.
  - When the BUSYD0 is detected low the host FW will switch the Voltage regulator to 1.8V, after which it instructs the SDMMC to start the timing critical section of the

Voltage Switch sequence by setting register bit VSWITCH. The hardware will continue to stop the SDMMC\_CK by holding it low for at least 5ms.

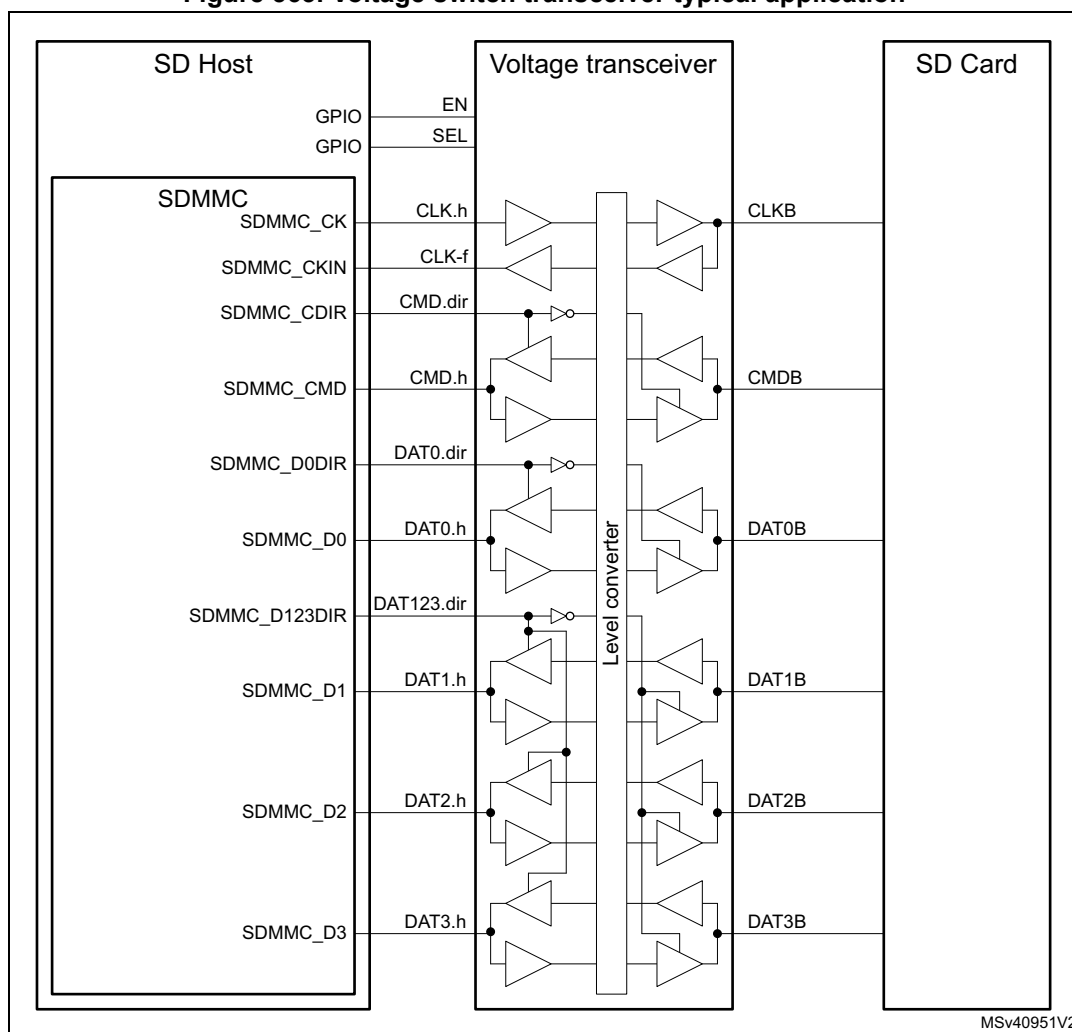
- When the BUSYD0 is detected high the host will abort the Voltage Switch sequence and cycle power the card.
6. The card after detecting SDMMC\_CK low will begin switching signaling voltage to 1.8V.
  7. The host SDMMC hardware after at least 5ms will restart the SDMMC\_CK.
  8. The card within 1ms from detecting SDMMC\_CK transition will drive CMD and DAT[3:0] high for at least 1 SDMMC\_CK cycle and then stop driving CMD and DAT[3:0].
  9. The host SDMMC hardware, 1ms after the SDMMC\_CK has been restarted, the SDMMC\_D0 is sampled into BUSYD0 and generate the VSWEND flag.
  10. The host, on the VSWEND flag, will check SDMMC\_D0 line using the BUSYD0 register bit, to confirm completion of voltage switch sequence:
    - When BUSYD0 is detected high, Voltage Switch has been completed successfully.
    - When BUSYD0 is detected low, Voltage Switch has failed, the host will cycle power the card power.

The minimum 5ms time to stop the SDMMC\_CK is derived from the internal ungated SDMMC\_CK clock, which will have a maximum frequency of 25MHz (SD mode), as set by the clock divider CLKDIV. The >5ms time will be counted by  $2^{12}$  cycles (10.24ms @ 400 kHz). If a lower SDMMC\_CK frequency is selected by the clock divider CLKDIV the time for the SDMMC\_CK clock to be stopped will be longer.

The maximum 1 ms time for the card to drive the SDMMC\_Dn and SDMMC\_CMD lines high is derived from the internal ungated SDMMC\_CK which will have a maximum frequency of 25MHz (SD mode), as set by the clock divider CLKDIV. The SDMMC will check the lines after >1ms time which will be counted by  $2^9$  cycles (1.28ms @25MHz). If a lower SDMMC\_CK frequency is selected by the clock divider CLKDIV the time to check the lines will be longer.

The signal voltage level is supported through an external voltage translation transceiver i.e. ST6G3244ME.

Figure 563. Voltage switch transceiver typical application



MSv40951V2

To interface with an external driver (a voltage switch transceiver), next to the standard signals the SDMMC uses the following signals:

**SDMMC\_CKIN** feedback input clock

**SDMMC\_CDIRE** I/O direction control for the CMD signal.

**SDMMC\_D0DIR** I/O direction control for the SDMMC\_D0 signal.

**SDMMC\_D123DIR** I/O direction control for the SDMMC\_D1, SDMMC\_D2 and SDMMC\_D3 signals.

The voltage transceiver signals **EN** and **SEL** are to be handled through general-purpose I/O.

The polarity of the SDMMC\_CDIRE, SDMMC\_D0DIR and SDMMC\_D123DIR signals can be selected through SDMMC\_POWER.DIRPOL control bit.



## 48.8 SDMMC registers

The device communicates to the system via 32-bit control registers accessible via AHB slave interface.

The peripheral registers have to be accessed by words (32-bit). Byte (8-bit) and halfword (16-bit) accesses generate an AHB bus error.

### 48.8.1 SDMMC power control register (SDMMC\_POWER)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIR POL	VSWI TCHEN	VSWI TCH	PWRCTRL[1:0]	
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **DIRPOL**: Data and command direction signals polarity selection

This bit can only be written when the SDMMC is in the power-off state (PWRCTRL = 00).

0: Voltage transceiver IOs driven as output when direction signal is low.

1: Voltage transceiver IOs driven as output when direction signal is high.

Bit 3 **VSWITCHEN**: Voltage switch procedure enable

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

This bit is used to stop the SDMMC\_CK after the voltage switch command response:

0: SDMMC\_CK clock kept unchanged after successfully received command response.

1: SDMMC\_CK clock stopped after successfully received command response.

Bit 2 **VSWITCH**: Voltage switch sequence start

This bit is used to start the timing critical section of the voltage switch sequence:

0: Voltage switch sequence not started and not active.

1: Voltage switch sequence started or active.

Bits 1:0 **PWRCTRL[1:0]**: SDMMC state control bits

These bits can only be written when the SDMMC is not in the power-on state (PWRCTRL ≠ 11).

These bits are used to define the functional state of the SDMMC signals:

00: After reset, Reset: the SDMMC is disabled and the clock to the Card is stopped, SDMMC\_D[7:0], and SDMMC\_CMD are HiZ and SDMMC\_CK is driven low.

When written 00, power-off: the SDMMC is disabled and the clock to the card is stopped, SDMMC\_D[7:0], SDMMC\_CMD and SDMMC\_CK are driven high.

01: Reserved. (When written 01, PWRCTRL value will not change)

10: Power-cycle, the SDMMC is disabled and the clock to the card is stopped, SDMMC\_D[7:0], SDMMC\_CMD and SDMMC\_CK are driven low.

11: Power-on: the card is clocked, The first 74 SDMMC\_CK cycles the SDMMC is still disabled. After the 74 cycles the SDMMC is enabled and the SDMMC\_D[7:0], SDMMC\_CMD and SDMMC\_CK are controlled according the SDMMC operation.

Any further write will be ignored, PWRCTRL value will keep 11.

## 48.8.2 SDMMC clock control register (SDMMC\_CLKCR)

Address offset: 0x004

Reset value: 0x0000 0000

The SDMMC\_CLKCR register controls the SDMMC\_CK output clock, the sdmmc\_rx\_ck receive clock, and the bus width.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]		BUS SPEED	DDR	HWFC_EN	NEG EDGE
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WID BUS[1:0]		Res.	PWR SAV	Res.	Res.	CLKDIV[9:0]									
rw	rw		rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **SELCLKRX[1:0]**: Receive clock selection

These bits can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: sdmmc\_io\_in\_ck selected as receive clock

01: SDMMC\_CKIN feedback clock selected as receive clock

10: Reserved

11: Reserved (select sdmmc\_io\_in\_ck)

Bit 19 **BUSPEED**: Bus speed mode selection between DS, HS, SDR12, SDR25 and SDR50, DDR50

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: DS, HS, SDR12, SDR25 bus speed mode selected

1: SDR50, DDR50 bus speed mode selected.

Bit 18 **DDR**: Data rate signaling selection

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

DDR rate shall only be selected with 4-bit or 8-bit wide bus mode. (WIDBUS > 00). DDR = 1 has no effect when WIDBUS = 00 (1-bit wide bus).

DDR rate shall only be selected with clock division >1. (CLKDIV > 0)

0: SDR Single data rate signaling

1: DDR double data rate signaling

Bit 17 **HWFC\_EN**: Hardware flow control enable

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: Hardware flow control is disabled

1: Hardware flow control is enabled

When Hardware flow control is enabled, the meaning of the TXFIFOE and RXFIFOE flags change, please see SDMMC status register definition in [Section 48.8.11](#).

Bit 16 **NEGEDGE**: SDMMC\_CK dephasing selection bit for data and command

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

When clock division = 1 (CLKDIV = 0), this bit has no effect. Data and Command change on SDMMC\_CK falling edge.

When clock division >1 (CLKDIV > 0) & DDR = 0:

- 0: - Command and data changed on the sdmmc\_ker\_ck falling edge succeeding the rising edge of SDMMC\_CK.
- SDMMC\_CK edge occurs on sdmmc\_ker\_ck rising edge.
- 1: - Command and data changed on the same sdmmc\_ker\_ck rising edge generating the SDMMC\_CK falling edge.

When clock division >1 (CLKDIV > 0) & DDR = 1:

- 0: - Command changed on the sdmmc\_ker\_ck falling edge succeeding the rising edge of SDMMC\_CK.
- Data changed on the sdmmc\_ker\_ck falling edge succeeding a SDMMC\_CK edge.
- SDMMC\_CK edge occurs on sdmmc\_ker\_ck rising edge.
- 1: - Command changed on the same sdmmc\_ker\_ck rising edge generating the SDMMC\_CK falling edge.
- Data changed on the SDMMC\_CK falling edge succeeding a SDMMC\_CK edge.
- SDMMC\_CK edge occurs on sdmmc\_ker\_ck rising edge.

Bits 15:14 **WIDBUS[1:0]**: Wide bus mode enable bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: Default 1-bit wide bus mode: SDMMC\_D0 used (Does not support DDR)

01: 4-bit wide bus mode: SDMMC\_D[3:0] used

10: 8-bit wide bus mode: SDMMC\_D[7:0] used

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PWRSAV**: Power saving configuration bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

For power saving, the SDMMC\_CK clock output can be disabled when the bus is idle by setting PWRSAV:

0: SDMMC\_CK clock is always enabled

1: SDMMC\_CK is only enabled when the bus is active

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **CLKDIV[9:0]**: Clock divide factor

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

This field defines the divide factor between the input clock (sdmmc\_ker\_ck) and the output clock (SDMMC\_CK):  $SDMMC\_CK\ frequency = sdmmc\_ker\_ck / [2 * CLKDIV]$ .

0x000: SDMMC\_CK frequency = sdmmc\_ker\_ck / 1 (Does not support DDR)

0x001: SDMMC\_CK frequency = sdmmc\_ker\_ck / 2

0x002: SDMMC\_CK frequency = sdmmc\_ker\_ck / 4

0x0XX: etc..

0x080: SDMMC\_CK frequency = sdmmc\_ker\_ck / 256

0xXXX: etc..

0x3FF: SDMMC\_CK frequency = sdmmc\_ker\_ck / 2046

- Note:
- 1 While the SD/SDIO card or eMMC is in identification mode, the SDMMC\_CK frequency must be less than 400 kHz.
  - 2 The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.
  - 3 At least seven sdmmc\_hclk clock periods are needed between two write accesses to this register. SDMMC\_CK can also be stopped during the ReadWait interval for SD I/O cards: in this case the SDMMC\_CLKCR register does not control SDMMC\_CK.

### 48.8.3 SDMMC argument register (SDMMC\_ARGR)

Address offset: 0x008

Reset value: 0x0000 0000

The SDMMC\_ARGR register contains a 32-bit command argument, which is sent to a card as part of a command message.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMDARG[31:16]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDARG[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **CMDARG[31:0]**: Command argument

These bits can only be written by firmware when CPSM is disabled (CPSMEN = 0). Command argument sent to a card as part of a command message. If a command contains an argument, it must be loaded into this register before writing a command to the command register.

### 48.8.4 SDMMC command register (SDMMC\_CMDR)

Address offset: 0x00C

Reset value: 0x0000 0000

The SDMMC\_CMDR register contains the command index and command type bits. The command index is sent to a card as part of a command message. The command type bits control the command path state machine (CPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMD SUS PEND
															r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOT EN	BOOT MODE	DT HOLD	CPSM EN	WAITP END	WAIT INT	WAITRESP[1:0]		CMD STOP	CMD TRANS	CMDINDEX[5:0]					
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:17 Reserved, must be kept at reset value.

- Bit 16 **CMDSPEND**: The CPSM treats the command as a Suspend or Resume command and signals interrupt period start/end  
This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).  
CMDSPEND = 1 and CMDTRANS = 0 Suspend command, start interrupt period when response bit BS=0.  
CMDSPEND = 1 and CMDTRANS = 1 Resume command with data, end interrupt period when response bit DF=1.
- Bit 15 **BOOTEN**: Enable boot mode procedure  
0: Boot mode procedure disabled  
1: Boot mode procedure enabled
- Bit 14 **BOOTMODE**: Select the boot mode procedure to be used  
This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0)  
0: Normal boot mode procedure selected  
1: Alternative boot mode procedure selected.
- Bit 13 **DTHOLD**: Hold new data block transmission and reception in the DPSM  
If this bit is set, the DPSM will not move from the Wait\_S state to the Send state or from the Wait\_R state to the Receive state.
- Bit 12 **CPSMEN**: Command path state machine (CPSM) enable bit  
This bit is written 1 by firmware, and cleared by hardware when the CPSM enters the Idle state.  
If this bit is set, the CPSM is enabled.  
When DTEN = 1, no command will be transferred nor boot procedure will be started.  
CPSMEN is cleared to 0.  
During ReadWait with SDMMC\_CK stopped no command will be sent and CPSMEN is kept 0.
- Bit 11 **WAITPEND**: CPSM waits for end of data transfer (CmdPend internal signal) from DPSM  
This bit when set, the CPSM waits for the end of data transfer trigger before it starts sending a command.  
WAITPEND is only taken into account when DTMODE = eMMC stream data transfer, WIDBUS = 1-bit wide bus mode, DPSSACT = 1 and DTDIR = from host to card.
- Bit 10 **WAITINT**: CPSM waits for interrupt request  
If this bit is set, the CPSM disables command timeout and waits for an card interrupt request (Response).  
If this bit is cleared in the CPSM Wait state, will cause the abort of the interrupt mode.
- Bits 9:8 **WAITRESP[1:0]**: Wait for response bits  
This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).  
They are used to configure whether the CPSM is to wait for a response, and if yes, which kind of response.  
00: No response, expect CMDSSENT flag  
01: Short response, expect CMDREND or CCRCFAIL flag  
10: Short response, expect CMDREND flag (No CRC)  
11: Long response, expect CMDREND or CCRCFAIL flag

Bit 7 **CMDSTOP**: The CPSM treats the command as a Stop Transmission command and signals Abort to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).  
If this bit is set, the CPSM issues the Abort signal to the DPSM when the command is sent.

Bit 6 **CMDTRANS**: The CPSM treats the command as a data transfer command, stops the interrupt period, and signals DataEnable to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).  
If this bit is set, the CPSM issues an end of interrupt period and issues DataEnable signal to the DPSM when the command is sent.

Bits 5:0 **CMDINDEX[5:0]**: Command index

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).  
The command index is sent to the card as part of a command message.

- Note:*
- 1 *At least seven sdmmc\_hclk clock periods are needed between two write accesses to this register.*
  - 2 *MultiMediaCard can send two kinds of response: short responses, 48 bits, or long responses, 136 bits. SD card and SD I/O card can send only short responses, the argument can vary according to the type of response: the software will distinguish the type of response according to the send command.*

### 48.8.5 SDMMC command response register (SDMMC\_RESPCMDR)

Address offset: 0x010

Reset value: 0x0000 0000

The SDMMC\_RESPCMDR register contains the command index field of the last command response received. If the command response transmission does not contain the command index field (long or OCR response), the RESPCMD field is unknown, although it must contain 111111b (the value of the reserved field from the response).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	RESPCMD[5:0]					
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RESPCMD[5:0]**: Response command index

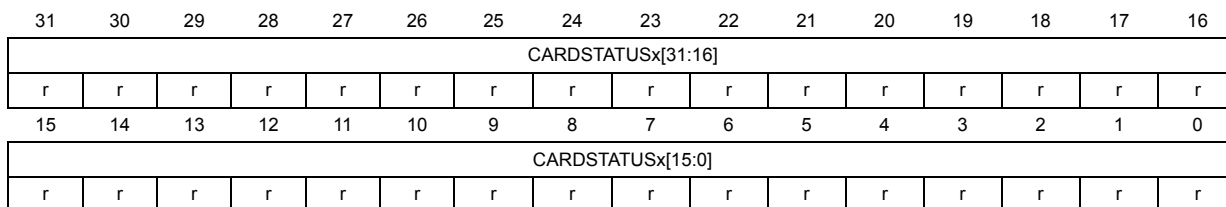
Read-only bit field. Contains the command index of the last command response received.

### 48.8.6 SDMMC response x register (SDMMC\_RESPxR)

Address offset: 0x010 + 0x004 \* x, (x = 1 to 4)

Reset value: 0x0000 0000

The SDMMC\_RESP1/2/3/4R registers contain the status of a card, which is part of the received response.



Bits 31:0 **CARDSTATUSx[31:0]**: see [Table 312](#)

The card status size is 32 or 128 bits, depending on the response type.

**Table 312. Response type and SDMMC\_RESPxR registers**

Register <sup>(1)</sup>	Short response	Long response
SDMMC_RESP1R	Card status[31:0]	Card status [127:96]
SDMMC_RESP2R	all 0	Card status [95:64]
SDMMC_RESP3R	all 0	Card status [63:32]
SDMMC_RESP4R	all 0	Card status [31:0] <sup>(2)</sup>

1. The most significant bit of the card status is received first.
2. The SDMMC\_RESP4R register LSB is always 0.

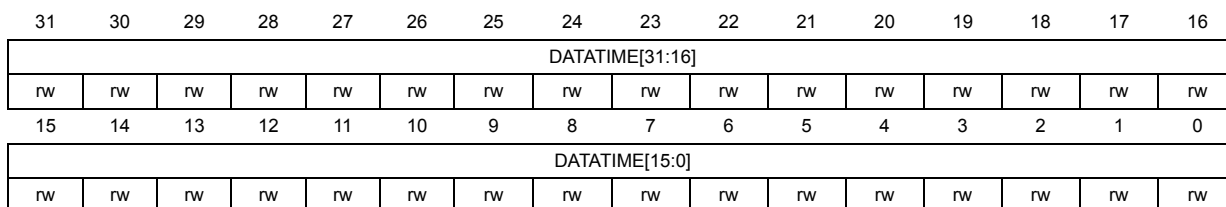
### 48.8.7 SDMMC data timer register (SDMMC\_DTIMER)

Address offset: 0x024

Reset value: 0x0000 0000

The SDMMC\_DTIMER register contains the data timeout period, in card bus clock periods.

A counter loads the value from the SDMMC\_DTIMER register, and starts decrementing when the data path state machine (DPSM) enters the Wait\_R or Busy state. If the timer reaches 0 while the DPSM is in either of these states, the timeout status flag is set.



Bits 31:0 **DATATIME[31:0]**: Data and R1b busy timeout period

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

Data and R1b busy timeout period expressed in card bus clock periods.

*Note:* A data transfer must be written to the data timer register and the data length register before being written to the data control register.

### 48.8.8 SDMMC data length register (SDMMC\_DLENR)

Address offset: 0x028

Reset value: 0x0000 0000

The SDMMC\_DLENR register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATALENGTH[24:16]								
							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
DATALENGTH[15:0]															
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 24:0 **DATALENGTH[24:0]**: Data length value

This register can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Number of data bytes to be transferred.

When DDR = 1 DATALENGTH is truncated to a multiple of 2. (The last odd byte is not transferred)

When DATALENGTH = 0 no data will be transferred, when requested by a CPSMEN and CMDTRANS = 1 also no command will be transferred. DTEN and CPSMEN are cleared to 0.

*Note:* For a block data transfer, the value in the data length register must be a multiple of the block size (see SDMMC\_DCTRL). A data transfer must be written to the data timer register and the data length register before being written to the data control register.

For an SDMMC multibyte transfer the value in the data length register must be between 1 and 512.



### 48.8.9 SDMMC data control register (SDMMC\_DCTRL)

Address offset: 0x02C

Reset value: 0x0000 0000

The SDMMC\_DCTRL register control the data path state machine (DPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FIFO RST	BOOT ACK EN	SDIO EN	RW MOD	RW STOP	RW START	DBLOCKSIZE[3:0]				DTMODE[1:0]		DTDIR	DTEN
		r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **FIFORST**: FIFO reset, will flush any remaining data

This bit can only be written by firmware when IDMAEN= 0 and DPSM is active (DPSMACT = 1). This bit will only take effect when a transfer error or transfer hold occurs.  
0: FIFO not affected.

1: Flush any remaining data and reset the FIFO pointers. This bit automatically will be cleared to 0 by hardware when DPSM gets inactive (DPSMACT = 0).

Bit 12 **BOOTACKEN**: Enable the reception of the boot acknowledgment

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Boot acknowledgment disabled, not expected to be received

1: Boot acknowledgment enabled, expected to be received

Bit 11 **SDIOEN**: SD I/O interrupt enable functions

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

If this bit is set, the DPSM enables the SD I/O card specific interrupt operation.

Bit 10 **RWMOD**: Read wait mode

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Read Wait control using SDMMC\_D2

1: Read Wait control stopping SDMMC\_CK

Bit 9 **RWSTOP**: Read wait stop

This bit is written by firmware and auto cleared by hardware when the DPSM moves from the READ\_WAIT state to the WAIT\_R or IDLE state.

0: No read wait stop.

1: Enable for read wait stop when DPSM is in the READ\_WAIT state.

Bit 8 **RWSTART**: Read wait start

If this bit is set, read wait operation starts.

**Bits 7:4 DBLOCKSIZE[3:0]:** Data block size

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Define the data block length when the block data transfer mode is selected:

0000: (0 decimal) lock length =  $2^0 = 1$  byte  
 0001: (1 decimal) lock length =  $2^1 = 2$  bytes  
 0010: (2 decimal) lock length =  $2^2 = 4$  bytes  
 0011: (3 decimal) lock length =  $2^3 = 8$  bytes  
 0100: (4 decimal) lock length =  $2^4 = 16$  bytes  
 0101: (5 decimal) lock length =  $2^5 = 32$  bytes  
 0110: (6 decimal) lock length =  $2^6 = 64$  bytes  
 0111: (7 decimal) lock length =  $2^7 = 128$  bytes  
 1000: (8 decimal) lock length =  $2^8 = 256$  bytes  
 1001: (9 decimal) lock length =  $2^9 = 512$  bytes  
 1010: (10 decimal) lock length =  $2^{10} = 1024$  bytes  
 1011: (11 decimal) lock length =  $2^{11} = 2048$  bytes  
 1100: (12 decimal) lock length =  $2^{12} = 4096$  bytes  
 1101: (13 decimal) lock length =  $2^{13} = 8192$  bytes  
 1110: (14 decimal) lock length =  $2^{14} = 16384$  bytes  
 1111: (15 decimal) reserved

When DATALENGTH is not a multiple of DBLOCKSIZE, the transferred data is truncated at a multiple of DBLOCKSIZE. (Any remain data will not be transferred.)

When DDR = 1, DBLOCKSIZE = 0000 shall not be used. (No data will be transferred)

**Bits 3:2 DTMODE[1:0]:** Data transfer mode selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

00: Block data transfer ending on block count.

01: SDIO multibyte data transfer.

10: eMMC Stream data transfer. (WIDBUS shall select 1-bit wide bus mode)

11: Block data transfer ending with STOP\_TRANSMISSION command (not to be used with DTEN initiated data transfers).

**Bit 1 DTDIR:** Data transfer direction selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: From host to card.

1: From card to host.

**Bit 0 DTEN:** Data transfer enable bit

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0). This bit is cleared by Hardware when data transfer completes.

This bit shall only be used to transfer data when no associated data transfer command is used, i.e. shall not be used with SD or eMMC cards.

0: Do not start data transfer without CPSM data transfer command.

1: Start data transfer without CPSM data transfer command.

**48.8.10 SDMMC data counter register (SDMMC\_DCNTR)**

Address offset: 0x030

Reset value: 0x0000 0000

The SDMMC\_DCNTR register loads the value from the data length register (see SDMMC\_DLENR) when the DPSM moves from the Idle state to the Wait\_R or Wait\_S state. As data is transferred, the counter decrements the value until it reaches 0. The DPSM then

moves to the Idle state and when there has been no error, the data status end flag (DATAEND) is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATACOUNT[24:16]								
							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
DATACOUNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATACOUNT[24:0]**: Data count value

When read, the number of remaining data bytes to be transferred is returned. Write has no effect.

*Note:* This register should be read only after the data transfer is complete, or hold. When reading after an error event the read data count value may be different from the real number of data bytes transferred.

### 48.8.11 SDMMC status register (SDMMC\_STAR)

Address offset: 0x034

Reset value: 0x0000 0000

The SDMMC\_STAR register is a read-only register. It contains two types of flag:

- Static flags (bits [28, 21, 11:0]): these bits remain asserted until they are cleared by writing to the SDMMC interrupt Clear register (see SDMMC\_ICR)
- Dynamic flags (bits [20:12]): these bits change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and de-asserted as data while written to the FIFO)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTC	IDMA TE	CK STOP	VSW END	ACK TIME OUT	ACK FAIL	SDIOIT	BUSY DOEND	BUSY D0	RX FIFOE	TX FIFOE	RX FIFO	TX FIFO
			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
RX FIFO HF	TX FIFO HE	CPSM ACT	DPSM ACT	DA BORT	DBCK END	DHOLD	DATA END	CMD SENT	CMDR END	RX OVERR	TX UNDER R	D TIME OUT	C TIME OUT	DCRC FAIL	CCRC FAIL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTC**: IDMA buffer transfer complete

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.

Bit 27 **IDMATE**: IDMA transfer error

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.

Bit 26 **CKSTOP**: SDMMC\_CK stopped in Voltage switch procedure

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.

- Bit 25 **VSWEND**: Voltage switch critical timing section completion  
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.
- Bit 24 **ACKTIMEOUT**: Boot acknowledgment timeout  
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.
- Bit 23 **ACKFAIL**: Boot acknowledgment received (boot acknowledgment check fail)  
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.
- Bit 22 **SDIOIT**: SDIO interrupt received  
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.
- Bit 21 **BUSYD0END**: end of SDMMC\_D0 Busy following a CMD response detected  
This indicates only end of busy following a CMD response. This bit does not signal busy due to data transfer. Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.  
0: card SDMMC\_D0 signal does NOT signal change from busy to not busy.  
1: card SDMMC\_D0 signal changed from busy to NOT busy.
- Bit 20 **BUSYD0**: Inverted value of SDMMC\_D0 line (Busy), sampled at the end of a CMD response and a second time 2 SDMMC\_CK cycles after the CMD response  
This bit is reset to not busy when the SDMMCD0 line changes from busy to not busy. This bit does not signal busy due to data transfer. This is a hardware status flag only, it does not generate an interrupt.  
0: card signals not busy on SDMMC\_D0.  
1: card signals busy on SDMMC\_D0.
- Bit 19 **RXFIFOE**: Receive FIFO empty  
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes full.
- Bit 18 **TXFIFOE**: Transmit FIFO empty  
This bit is cleared when one FIFO location becomes full.
- Bit 17 **RXFIFO**: Receive FIFO full  
This bit is cleared when one FIFO location becomes empty.
- Bit 16 **TXFIFO**: Transmit FIFO full  
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes empty.
- Bit 15 **RXFIFOHF**: Receive FIFO half full  
There are at least half the number of words in the FIFO. This bit is cleared when the FIFO becomes half+1 empty.
- Bit 14 **TXFIFOHE**: Transmit FIFO half empty  
At least half the number of words can be written into the FIFO. This bit is cleared when the FIFO becomes half+1 full.
- Bit 13 **CPSMACT**: Command path state machine active, i.e. not in Idle state  
This is a hardware status flag only, does not generate an interrupt.
- Bit 12 **DPSMACT**: Data path state machine active, i.e. not in Idle state  
This is a hardware status flag only, does not generate an interrupt.
- Bit 11 **DABORT**: Data transfer aborted by CMD12  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.

- Bit 10 **DBCKEND**: Data block sent/received  
(CRC check passed) and DPSM moves to the READWAIT state. Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.
- Bit 9 **DHOLD**: Data transfer Hold  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.
- Bit 8 **DATAEND**: Data transfer ended correctly  
(data counter, DATACOUNT is zero and no errors occur). Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.
- Bit 7 **CMDSENT**: Command sent (no response required)  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.
- Bit 6 **CMDREND**: Command response received (CRC check passed, or no CRC)  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.
- Bit 5 **RXOVERR**: Received FIFO overrun error  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.
- Bit 4 **TXUNDERR**: Transmit FIFO underrun error  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.
- Bit 3 **DTIMEOUT**: Data timeout  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.
- Bit 2 **CTIMEOUT**: Command response timeout  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.  
The Command Timeout period has a fixed value of 64 SDMMC\_CK clock periods.
- Bit 1 **DCRCFAIL**: Data block sent/received (CRC check failed)  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.
- Bit 0 **CCRCFAIL**: Command response received (CRC check failed)  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC\_ICR.

*Note:* FIFO interrupt flags shall be masked in SDMMC\_MASKR when using IDMA mode.

### 48.8.12 SDMMC interrupt clear register (SDMMC\_ICR)

Address offset: 0x038

Reset value: 0x0000 0000

The SDMMC\_ICR register is a write-only register. Writing a bit with 1 clears the corresponding bit in the SDMMC\_STAR status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCC	IDMA TEC	CK STOP C	VSW ENDC	ACK TIME OUTC	ACK FAILC	SDIO ITC	BUSY D0 ENDC	Res.	Res.	Res.	Res.	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
Res.	Res.	Res.	Res.	D ABOR TC	DBCK ENDC	DHOLD C	DATA ENDC	CMD SENTC	CMDR ENDC	RX OVERR C	TX UNDER RC	D TIME OUTC	C TIME OUTC	DCRC FAILC	CCRC FAILC
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCC**: IDMA buffer transfer complete clear bit  
Set by software to clear the IDMABTC flag.  
0: IDMABTC not cleared  
1: IDMABTC cleared

Bit 27 **IDMATEC**: IDMA transfer error clear bit  
Set by software to clear the IDMATE flag.  
0: IDMATE not cleared  
1: IDMATE cleared

Bit 26 **CKSTOPC**: CKSTOP flag clear bit  
Set by software to clear the CKSTOP flag.  
0: CKSTOP not cleared  
1: CKSTOP cleared

Bit 25 **VSWENDC**: VSWEND flag clear bit  
Set by software to clear the VSWEND flag.  
0: VSWEND not cleared  
1: VSWEND cleared

Bit 24 **ACKTIMEOUTC**: ACKTIMEOUT flag clear bit  
Set by software to clear the ACKTIMEOUT flag.  
0: ACKTIMEOUT not cleared  
1: ACKTIMEOUT cleared

Bit 23 **ACKFAILC**: ACKFAIL flag clear bit  
Set by software to clear the ACKFAIL flag.  
0: ACKFAIL not cleared  
1: ACKFAIL cleared

Bit 22 **SDIOITC**: SDIOIT flag clear bit  
Set by software to clear the SDIOIT flag.  
0: SDIOIT not cleared  
1: SDIOIT cleared

Bit 21 **BUSYD0ENDC**: BUSYD0END flag clear bit  
Set by software to clear the BUSYD0END flag.  
0: BUSYD0END not cleared  
1: BUSYD0END cleared

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **DABORTC**: DABORT flag clear bit  
Set by software to clear the DABORT flag.  
0: DABORT not cleared  
1: DABORT cleared

Bit 10 **DBCKENDC**: DBCKEND flag clear bit  
Set by software to clear the DBCKEND flag.  
0: DBCKEND not cleared  
1: DBCKEND cleared

Bit 9 **DHOLDC**: DHOLD flag clear bit  
Set by software to clear the DHOLD flag.  
0: DHOLD not cleared  
1: DHOLD cleared

- Bit 8 **DATAENDC**: DATAEND flag clear bit  
Set by software to clear the DATAEND flag.  
0: DATAEND not cleared  
1: DATAEND cleared
- Bit 7 **CMDSENTC**: CMDSSENT flag clear bit  
Set by software to clear the CMDSSENT flag.  
0: CMDSSENT not cleared  
1: CMDSSENT cleared
- Bit 6 **CMDREND**: CMDREND flag clear bit  
Set by software to clear the CMDREND flag.  
0: CMDREND not cleared  
1: CMDREND cleared
- Bit 5 **RXOVERRC**: RXOVERR flag clear bit  
Set by software to clear the RXOVERR flag.  
0: RXOVERR not cleared  
1: RXOVERR cleared
- Bit 4 **TXUNDERRC**: TXUNDERR flag clear bit  
Set by software to clear TXUNDERR flag.  
0: TXUNDERR not cleared  
1: TXUNDERR cleared
- Bit 3 **DTIMEOUTC**: DTIMEOUT flag clear bit  
Set by software to clear the DTIMEOUT flag.  
0: DTIMEOUT not cleared  
1: DTIMEOUT cleared
- Bit 2 **CTIMEOUTC**: CTIMEOUT flag clear bit  
Set by software to clear the CTIMEOUT flag.  
0: CTIMEOUT not cleared  
1: CTIMEOUT cleared
- Bit 1 **DCRCFAILC**: DCRCFAIL flag clear bit  
Set by software to clear the DCRCFAIL flag.  
0: DCRCFAIL not cleared  
1: DCRCFAIL cleared
- Bit 0 **CCRCFAILC**: CCRCFAIL flag clear bit  
Set by software to clear the CCRCFAIL flag.  
0: CCRCFAIL not cleared  
1: CCRCFAIL cleared

### 48.8.13 SDMMC mask register (SDMMC\_MASKR)

Address offset: 0x03C

Reset value: 0x0000 0000

The interrupt mask register determines which status flags generate an interrupt request by setting the corresponding bit to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCIE	Res.	CK STOP IE	VSW ENDIE	ACK TIME OUTIE	ACK FAILIE	SDIO ITIE	BUSY DO ENDIE	Res.	Res.	TX FIFO EIE	RX FIFO FIE	Res.
			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
RX FIFO HFIE	TX FIFO HEIE	Res.	Res.	DA BORT IE	DBCK ENDIE	DHOLD IE	DATA ENDIE	CMD SENT IE	CMDR ENDIE	RX OVERR IE	TX UNDER RIE	D TIME OUTIE	C TIME OUTIE	DCRC FAILIE	CCRC FAILIE
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCIE**: IDMA buffer transfer complete interrupt enable

Set and cleared by software to enable/disable the interrupt generated when the IDMA has transferred all data belonging to a memory buffer.

0: IDMA buffer transfer complete interrupt disabled

1: IDMA buffer transfer complete interrupt enabled

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CKSTOPIE**: Voltage Switch clock stopped interrupt enable

Set and cleared by software to enable/disable interrupt caused by Voltage Switch clock stopped.

0: Voltage Switch clock stopped interrupt disabled

1: Voltage Switch clock stopped interrupt enabled

Bit 25 **VSWENDIE**: Voltage switch critical timing section completion interrupt enable

Set and cleared by software to enable/disable the interrupt generated when voltage switch critical timing section completion.

0: Voltage switch critical timing section completion interrupt disabled

1: Voltage switch critical timing section completion interrupt enabled

Bit 24 **ACKTIMEOUTIE**: Acknowledgment timeout interrupt enable

Set and cleared by software to enable/disable interrupt caused by acknowledgment timeout.

0: Acknowledgment timeout interrupt disabled

1: Acknowledgment timeout interrupt enabled

Bit 23 **ACKFAILIE**: Acknowledgment Fail interrupt enable

Set and cleared by software to enable/disable interrupt caused by acknowledgment Fail.

0: Acknowledgment Fail interrupt disabled

1: Acknowledgment Fail interrupt enabled

Bit 22 **SDIOITIE**: SDIO mode interrupt received interrupt enable

Set and cleared by software to enable/disable the interrupt generated when receiving the SDIO mode interrupt.

0: SDIO Mode interrupt received interrupt disabled

1: SDIO Mode interrupt received interrupt enabled



- Bit 21 **BUSYD0ENDIE**: BUSYD0END interrupt enable  
Set and cleared by software to enable/disable the interrupt generated when SDMMC\_D0 signal changes from busy to NOT busy following a CMD response.  
0: BUSYD0END interrupt disabled  
1: BUSYD0END interrupt enabled
- Bits 20:19 Reserved, must be kept at reset value.
- Bit 18 **TXFIFOEIE**: Tx FIFO empty interrupt enable  
Set and cleared by software to enable/disable interrupt caused by Tx FIFO empty.  
0: Tx FIFO empty interrupt disabled  
1: Tx FIFO empty interrupt enabled
- Bit 17 **RXFIFOFIE**: Rx FIFO full interrupt enable  
Set and cleared by software to enable/disable interrupt caused by Rx FIFO full.  
0: Rx FIFO full interrupt disabled  
1: Rx FIFO full interrupt enabled
- Bit 16 Reserved, must be kept at reset value.
- Bit 15 **RXFIFOHFIE**: Rx FIFO half full interrupt enable  
Set and cleared by software to enable/disable interrupt caused by Rx FIFO half full.  
0: Rx FIFO half full interrupt disabled  
1: Rx FIFO half full interrupt enabled
- Bit 14 **TXFIFOHEIE**: Tx FIFO half empty interrupt enable  
Set and cleared by software to enable/disable interrupt caused by Tx FIFO half empty.  
0: Tx FIFO half empty interrupt disabled  
1: Tx FIFO half empty interrupt enabled
- Bits 13:12 Reserved, must be kept at reset value.
- Bit 11 **DABORTIE**: Data transfer aborted interrupt enable  
Set and cleared by software to enable/disable interrupt caused by a data transfer being aborted.  
0: Data transfer abort interrupt disabled  
1: Data transfer abort interrupt enabled
- Bit 10 **DBCKENDIE**: Data block end interrupt enable  
Set and cleared by software to enable/disable interrupt caused by data block end.  
0: Data block end interrupt disabled  
1: Data block end interrupt enabled
- Bit 9 **DHOLDIE**: Data hold interrupt enable  
Set and cleared by software to enable/disable the interrupt generated when sending new data is hold in the DPSM Wait\_S state.  
0: Data hold interrupt disabled  
1: Data hold interrupt enabled
- Bit 8 **DATAENDIE**: Data end interrupt enable  
Set and cleared by software to enable/disable interrupt caused by data end.  
0: Data end interrupt disabled  
1: Data end interrupt enabled
- Bit 7 **CMDSSENTIE**: Command sent interrupt enable  
Set and cleared by software to enable/disable interrupt caused by sending command.  
0: Command sent interrupt disabled  
1: Command sent interrupt enabled

- Bit 6 **CMDRENDIE**: Command response received interrupt enable  
 Set and cleared by software to enable/disable interrupt caused by receiving command response.  
 0: Command response received interrupt disabled  
 1: command Response received interrupt enabled
- Bit 5 **RXOVERRIE**: Rx FIFO overrun error interrupt enable  
 Set and cleared by software to enable/disable interrupt caused by Rx FIFO overrun error.  
 0: Rx FIFO overrun error interrupt disabled  
 1: Rx FIFO overrun error interrupt enabled
- Bit 4 **TXUNDERRIE**: Tx FIFO underrun error interrupt enable  
 Set and cleared by software to enable/disable interrupt caused by Tx FIFO underrun error.  
 0: Tx FIFO underrun error interrupt disabled  
 1: Tx FIFO underrun error interrupt enabled
- Bit 3 **DTIMEOUTIE**: Data timeout interrupt enable  
 Set and cleared by software to enable/disable interrupt caused by data timeout.  
 0: Data timeout interrupt disabled  
 1: Data timeout interrupt enabled
- Bit 2 **CTIMEOUTIE**: Command timeout interrupt enable  
 Set and cleared by software to enable/disable interrupt caused by command timeout.  
 0: Command timeout interrupt disabled  
 1: Command timeout interrupt enabled
- Bit 1 **DCRCFAILIE**: Data CRC fail interrupt enable  
 Set and cleared by software to enable/disable interrupt caused by data CRC failure.  
 0: Data CRC fail interrupt disabled  
 1: Data CRC fail interrupt enabled
- Bit 0 **CCRCFAILIE**: Command CRC fail interrupt enable  
 Set and cleared by software to enable/disable interrupt caused by command CRC failure.  
 0: Command CRC fail interrupt disabled  
 1: Command CRC fail interrupt enabled

#### 48.8.14 SDMMC acknowledgment timer register (SDMMC\_ACKTIMER)

Address offset: 0x040

Reset value: 0x0000 0000

The SDMMC\_ACKTIMER register contains the acknowledgment timeout period, in SDMMC\_CK bus clock periods.

A counter loads the value from the SDMMC\_ACKTIMER register, and starts decrementing when the data path state machine (DPSM) enters the Wait\_Ack state. If the timer reaches 0 while the DPSM is in this states, the acknowledgment timeout status flag is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16			
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACKTIME[24:16]											
							r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
ACKTIME[15:0]																		
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w			



Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **ACKTIME[24:0]**: Boot acknowledgment timeout period

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).  
 Boot acknowledgment timeout period expressed in card bus clock periods.

*Note:* The data transfer must be written to the acknowledgment timer register before being written to the data control register.

### 48.8.15 SDMMC data FIFO register (SDMMC\_FIFOR)

Address offset: 0x080 to 0x0BC

Reset value: 0x0000 0000

The receive and transmit FIFOs can be only read or written as word (32-bit) wide registers. The FIFOs contain 16 entries on sequential addresses. This allows the CPU to use its load and store multiple operands to read from/write to the FIFO.

When accessing SDMMC\_FIFOR with half word or byte access an AHB bus fault is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIFODATA[31:16]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIFODATA[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **FIFODATA[31:0]**: Receive and transmit FIFO data

This register can only be read or written by firmware when the DPSM is active (DPSMACT = 1).  
 The FIFO data occupies 16 entries of 32-bit words.

### 48.8.16 SDMMC DMA control register (SDMMC\_IDMACTRLR)

Address offset: 0x050

Reset value: 0x0000 0000

The receive and transmit FIFOs can be read or written as 32-bit wide registers. The FIFOs contain 32 entries on 32 sequential addresses. This allows the CPU to use its load and store multiple operands to read from/write to the FIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	IDMAB ACT	IDMAB MODE	IDMA EN
													r/w	r/w	r/w



Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **IDMABACT**: Double buffer mode active buffer indication

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0). When IDMA is enabled this bit is toggled by hardware.

0: When IDMA is enabled, uses buffer0 and firmware write access to IDMABASE0 is prohibited.

1: When IDMA is enabled, uses buffer1 and firmware write access to IDMABASE1 is prohibited.

Bit 1 **IDMABMODE**: Buffer mode selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Single buffer mode.

1: Double buffer mode.

Bit 0 **IDMAEN**: IDMA enable

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: IDMA disabled

1: IDMA enabled

### 48.8.17 SDMMC IDMA buffer size register (SDMMC\_IDMABSIZER)

Address offset: 0x054

Reset value: 0x0000 0000

The SDMMC\_IDMABSIZER register contains the buffers size when in double buffer configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	IDMABNDT[7:0]								Res.	Res.	Res.	Res.	Res.
			rw	rw	rw	rw	rw	rw	rw	rw					

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:5 **IDMABNDT[7:0]**: Number of bytes per buffer

This 8-bit value shall be multiplied by 8 to get the size of the buffer in 32-bit words and by 32 to get the size of the buffer in bytes.

Example: IDMABNDT = 0x01: buffer size = 8 words = 32 bytes.

Example: IDMABNDT = 0x80: buffer size = 1024 words = 4 Kbyte

These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 4:0 Reserved, must be kept at reset value.

### 48.8.18 SDMMC IDMA buffer 0 base address register (SDMMC\_IDMABASE0R)

Address offset: 0x058

Reset value: 0x0000 0000

The SDMMC\_IDMABASE0R register contains the memory buffer base address in single buffer configuration and the buffer 0 base address in double buffer configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDMABASE0[31:16]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDMABASE0[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r	r

Bits 31:0 **IDMABASE0[31:0]**: Buffer 0 memory base address bits [31:2], shall be word aligned (bit [1:0] are always 0 and read only)

This register can be written by firmware when DPSM is inactive (DPSMACT = 0), and can dynamically be written by firmware when DPSM active (DPSMACT = 1) and memory buffer 0 is inactive (IDMABACT = '1').

### 48.8.19 SDMMC IDMA buffer 1 base address register (SDMMC\_IDMABASE1R)

Address offset: 0x05C

Reset value: 0x0000 0000

The SDMMC\_IDMABASE1R register contains the double buffer configuration second buffer memory base address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDMABASE1[31:16]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDMABASE1[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r	r

Bits 31:0 **IDMABASE1[31:0]**: Buffer 1 memory base address, shall be word aligned (bit [1:0] are always 0 and read only)

This register can be written by firmware when DPSM is inactive (DPSMACT = 0), and can dynamically be written by firmware when DPSM active (DPSMACT = 1) and memory buffer 1 is inactive (IDMABACT = '0').

### 48.8.20 SDMMC register map

The following table summarizes the SDMMC registers.

**Table 313. SDMMC register map**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0x000	SDMMC_POWER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRPOL	VSWITCHEN	VSWITCH		PWRCTRL[1:0]	
	Reset value																														0	0	0	0	0
0x004	SDMMC_CLKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]	BUSPEED	DDR	HWFC_EN	NEGEDGE	WIDBUS[1:0]			Res.	PWRSVAV	Res.	Res.												CLKDIV[9:0]
	Reset value											0	0	0	0	0	0	0	0			0			0	0	0	0	0	0	0	0	0	0	0
0x008	SDMMC_ARGR	CMDARG[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	SDMMC_CMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDSPEND	BOOTEN	BOOTMODE	DTHOLD	CPSMEN	WAITPEND	WAITINT	WAITRESP[1:0]	CMDSTOP	CMDTRANS								CMDINDEX[5:0]	
	Reset value																																		
0x010	SDMMC_RESPCMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESPCMD[5:0]	
	Reset value																																		0
0x014	SDMMC_RESP1R	CARDSTATUS1[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	
0x018	SDMMC_RESP2R	CARDSTATUS2[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	
0x01C	SDMMC_RESP3R	CARDSTATUS3[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	
0x020	SDMMC_RESP4R	CARDSTATUS4[31:0]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x024	SDMMC_DTIMER	DATATIME[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	
0x028	SDMMC_DLENR	Res.	Res.	Res.	Res.	Res.	Res.																												DATALENGTH[24:0]
	Reset value																																		0

Table 313. SDMMC register map (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				
0x02C	<b>SDMMC_DCTRLR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFORST	BOOTACKEN	SIOEN	RWMOD	RWSTOP	RWSTART	DBLOCK SIZE[3:0]			DTMODE	DTDJR	DTEN						
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x030	<b>SDMMC_DCNTR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATACOUNT[24:0]																												
	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				
0x034	<b>SDMMC_STAR</b>	Res.	Res.	Res.	IDMABTC	IDMATE	CKSTOP	VSWEND	ACKTIMEOUT	ACKFAIL	SDIOIT	BUSYD0END	BUSYD0	RXFIFOE	TXFIFOE	RXFIFOE	TXFIFOE	RXFIFOE	TXFIFOE	CPSMACT	DPSMACT	DABORT	DBCKEND	DHOLD	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL				
	Reset value				0	0	0	0	0	0	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	<b>SDMMC_ICR</b>	Res.	Res.	Res.	IDMABTCC	IDMATEC	CKSTOPC	VSWENDC	ACKTIMEOUTC	ACKFAILC	SDIOITC	BUSYD0ENDC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DABORTC	DBCKENDC	DHOLDC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC				
	Reset value				0	0	0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0			
0x03C	<b>SDMMC_MASKR</b>	Res.	Res.	Res.	IDMABTCIE	Res.	CKSTOPIE	VSWENDIE	ACKTIMEOUTIE	ACKFAILIE	SDIOITIE	BUSYD0ENDIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DABORTIE	DBCKENDIE	DHOLDIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE				
	Reset value				0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0			
0x040	<b>SDMMC_ACKTIMER</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACKTIME[24:0]																												
	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			
0x044 - 0x04C	<b>Reserved</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
0x050	<b>SDMMC_IDMACTRLR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
	Reset value																														0	0	0	0			
0x054	<b>SDMMC_IDMABSIZER</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABNDT[7:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																					0	0	0	0	0	0	0	0	0							
0x058	<b>SDMMC_IDMABASE0R</b>	IDMABASE0[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			
0x05C	<b>SDMMC_IDMABASE1R</b>	IDMABASE1[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			
0x060 - 0x07C	<b>Reserved</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			



Table 313. SDMMC register map (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
0x080	SDMMC_FIFOR	FIFODATA[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
0x084 - x3FC	Reserved	Res.																															

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.



## 49 Controller area network (bxCAN)

### 49.1 Introduction

The **Basic Extended CAN** peripheral, named **bxCAN**, interfaces the CAN network. It supports the CAN protocols version 2.0A and B. It has been designed to manage a high number of incoming messages efficiently with a minimum CPU load. It also meets the priority requirements for transmit messages.

For safety-critical applications, the CAN controller provides all hardware functions for supporting the CAN Time Triggered Communication option.

### 49.2 bxCAN main features

- Supports CAN protocol version 2.0 A, B Active
- Bit rates up to 1 Mbit/s
- Supports the Time Triggered Communication option

#### Transmission

- Three transmit mailboxes
- Configurable transmit priority
- Time Stamp on SOF transmission

#### Reception

- Two receive FIFOs with three stages
- Scalable filter banks:
  - 28 filter banks shared between CAN1 and CAN2 for dual CAN
- Identifier list feature
- Configurable FIFO overrun
- Time Stamp on SOF reception

#### Time-triggered communication option

- Disable automatic retransmission mode
- 16-bit free running timer
- Time Stamp sent in last two data bytes

#### Management

- Maskable interrupts
- Software-efficient mailbox mapping at a unique address space

#### Dual CAN peripheral configuration

- CAN1: Master bxCAN for managing the communication between a Slave bxCAN and the 512-byte SRAM memory
- CAN2: Slave bxCAN, with no direct access to the SRAM memory.
- The two bxCAN cells share the 512-byte SRAM memory (see [Figure 565: Dual-CAN block diagram](#))

### 49.3 bxCAN general description

In today CAN applications, the number of nodes in a network is increasing and often several networks are linked together via gateways. Typically the number of messages in the system (to be handled by each node) has significantly increased. In addition to the application messages, Network Management and Diagnostic messages have been introduced.

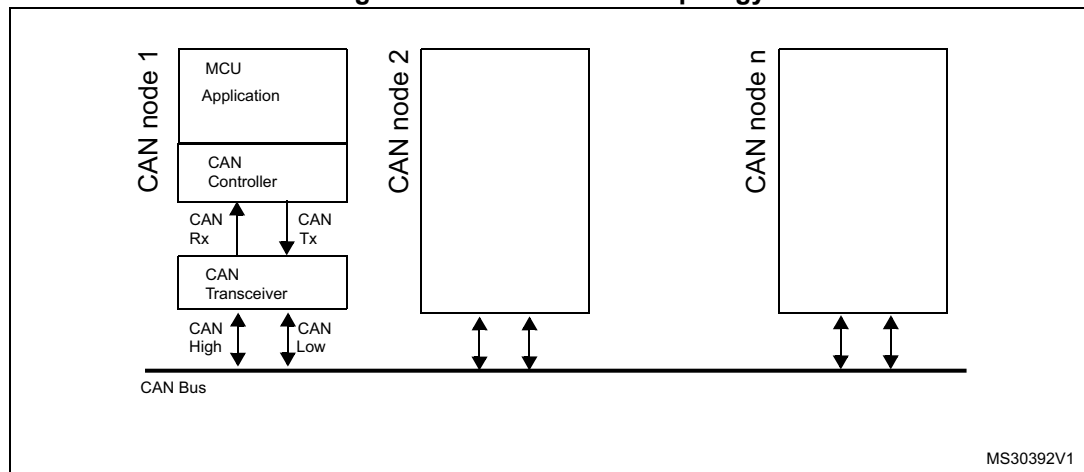
- An enhanced filtering mechanism is required to handle each type of message.

Furthermore, application tasks require more CPU time, therefore real-time constraints caused by message reception have to be reduced.

- A receive FIFO scheme allows the CPU to be dedicated to application tasks for a long time period without losing messages.

The standard HLP (Higher Layer Protocol) based on standard CAN drivers requires an efficient interface to the CAN controller.

Figure 564. CAN network topology



#### 49.3.1 CAN 2.0B active core

The bxCAN module handles the transmission and the reception of CAN messages fully autonomously. Standard identifiers (11-bit) and extended identifiers (29-bit) are fully supported by hardware.

#### 49.3.2 Control, status and configuration registers

The application uses these registers to:

- Configure CAN parameters, e.g. baud rate
- Request transmissions
- Handle receptions
- Manage interrupts
- Get diagnostic information

#### 49.3.3 Tx mailboxes

Three transmit mailboxes are provided to the software for setting up messages. The transmission Scheduler decides which mailbox has to be transmitted first.

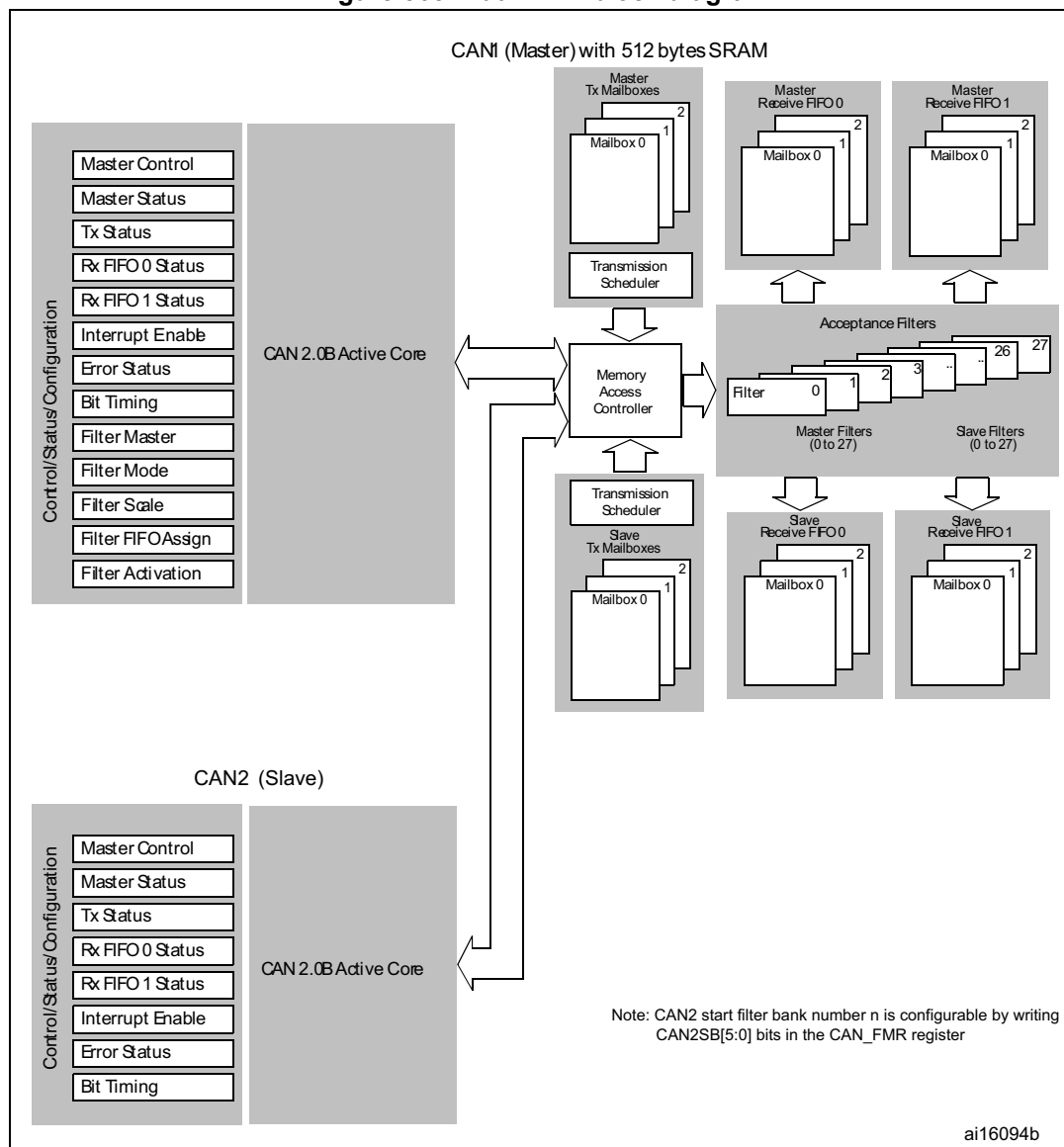
### 49.3.4 Acceptance filters

The bxCAN provides up to 28 scalable/configurable identifier filter banks in dual CAN configuration, for selecting the incoming messages, that the software needs and discarding the others.

#### Receive FIFO

Two receive FIFOs are used by hardware to store the incoming messages. Three complete messages can be stored in each FIFO. The FIFOs are managed completely by hardware.

Figure 565. Dual-CAN block diagram



## 49.4 bxCAN operating modes

bxCAN has three main operating modes: **initialization**, **normal** and **Sleep**. After a hardware reset, bxCAN is in Sleep mode to reduce power consumption and an internal pull-up is active on CANTX. The software requests bxCAN to enter **initialization** or **Sleep** mode by setting the INRQ or SLEEP bits in the CAN\_MCR register. Once the mode has been entered, bxCAN confirms it by setting the INAK or SLAK bits in the CAN\_MSR register and the internal pull-up is disabled. When neither INAK nor SLAK are set, bxCAN is in **normal** mode. Before entering **normal** mode bxCAN always has to **synchronize** on the CAN bus. To synchronize, bxCAN waits until the CAN bus is idle, this means 11 consecutive recessive bits have been monitored on CANRX.

### 49.4.1 Initialization mode

The software initialization can be done while the hardware is in Initialization mode. To enter this mode the software sets the INRQ bit in the CAN\_MCR register and waits until the hardware has confirmed the request by setting the INAK bit in the CAN\_MSR register.

To leave Initialization mode, the software clears the INQR bit. bxCAN has left Initialization mode once the INAK bit has been cleared by hardware.

While in Initialization Mode, all message transfers to and from the CAN bus are stopped and the status of the CAN bus output CANTX is recessive (high).

Entering Initialization Mode does not change any of the configuration registers.

To initialize the CAN Controller, software has to set up the Bit Timing (CAN\_BTR) and CAN options (CAN\_MCR) registers.

To initialize the registers associated with the CAN filter banks (mode, scale, FIFO assignment, activation and filter values), software has to set the FINIT bit (CAN\_FMR). Filter initialization also can be done outside the initialization mode.

*Note:* When FINIT=1, CAN reception is deactivated.

*The filter values also can be modified by deactivating the associated filter activation bits (in the CAN\_FA1R register).*

*If a filter bank is not used, it is recommended to leave it non active (leave the corresponding FACT bit cleared).*

### 49.4.2 Normal mode

Once the initialization is complete, the software must request the hardware to enter Normal mode to be able to synchronize on the CAN bus and start reception and transmission.

The request to enter Normal mode is issued by clearing the INRQ bit in the CAN\_MCR register. The bxCAN enters Normal mode and is ready to take part in bus activities when it has synchronized with the data transfer on the CAN bus. This is done by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus Idle state). The switch to Normal mode is confirmed by the hardware by clearing the INAK bit in the CAN\_MSR register.

The initialization of the filter values is independent from Initialization Mode but must be done while the filter is not active (corresponding FACTx bit cleared). The filter scale and mode configuration must be configured before entering Normal Mode.

### 49.4.3 Sleep mode (low-power)

To reduce power consumption, bxCAN has a low-power mode called Sleep mode. This mode is entered on software request by setting the SLEEP bit in the CAN\_MCR register. In this mode, the bxCAN clock is stopped, however software can still access the bxCAN mailboxes.

If software requests entry to **initialization** mode by setting the INRQ bit while bxCAN is in **Sleep** mode, it must also clear the SLEEP bit.

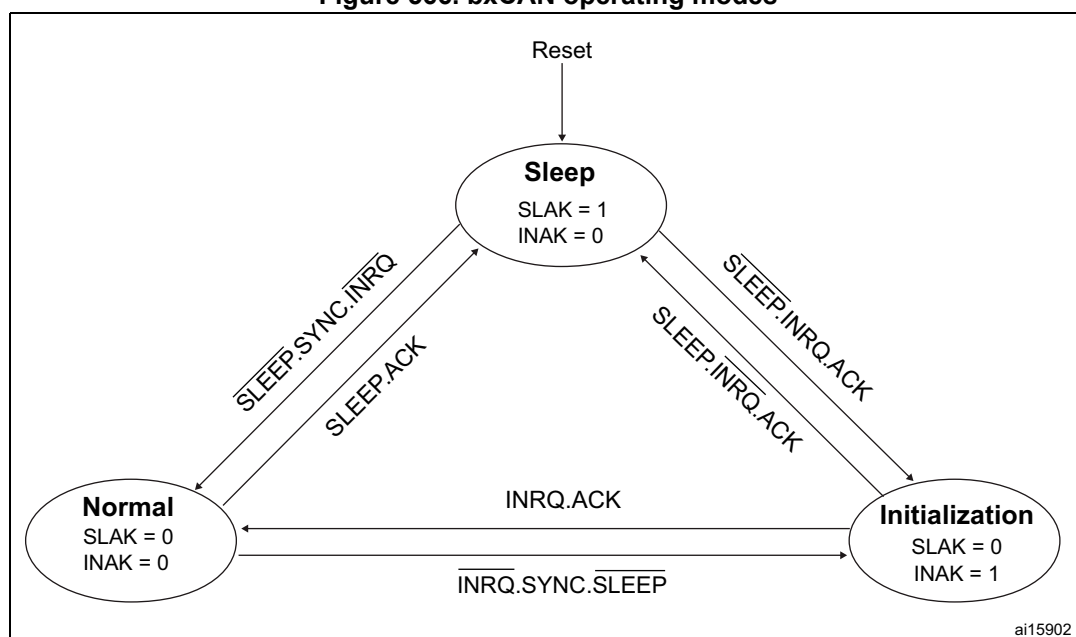
bxCAN can be woken up (exit Sleep mode) either by software clearing the SLEEP bit or on detection of CAN bus activity.

On CAN bus activity detection, hardware automatically performs the wakeup sequence by clearing the SLEEP bit if the AWUM bit in the CAN\_MCR register is set. If the AWUM bit is cleared, software has to clear the SLEEP bit when a wakeup interrupt occurs, in order to exit from Sleep mode.

*Note: If the wakeup interrupt is enabled (WKUIE bit set in CAN\_IER register) a wakeup interrupt will be generated on detection of CAN bus activity, even if the bxCAN automatically performs the wakeup sequence.*

After the SLEEP bit has been cleared, Sleep mode is exited once bxCAN has synchronized with the CAN bus, refer to [Figure 566: bxCAN operating modes](#). The Sleep mode is exited once the SLAK bit has been cleared by hardware.

Figure 566. bxCAN operating modes



1. ACK = The wait state during which hardware confirms a request by setting the INAK or SLAK bits in the CAN\_MSR register
2. SYNC = The state during which bxCAN waits until the CAN bus is idle, meaning 11 consecutive recessive bits have been monitored on CANRX

## 49.5 Test mode

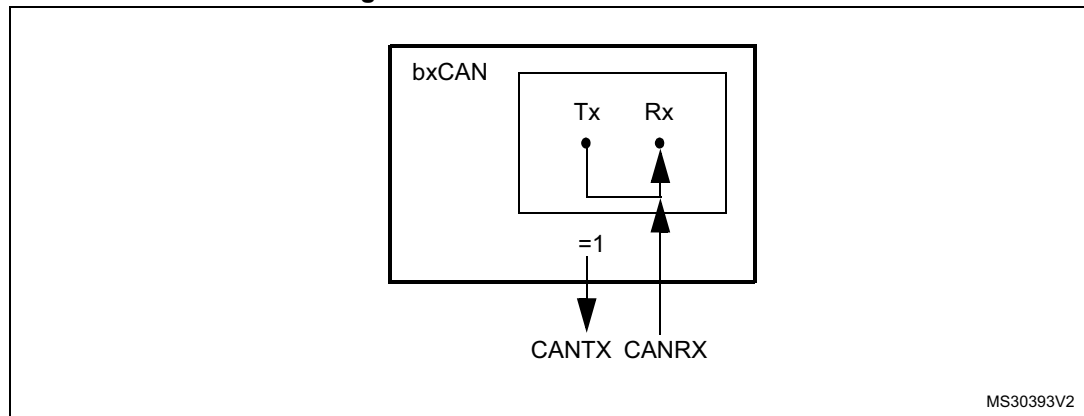
Test mode can be selected by the SILM and LBKM bits in the CAN\_BTR register. These bits must be configured while bxCAN is in Initialization mode. Once test mode has been selected, the INRQ bit in the CAN\_MCR register must be reset to enter Normal mode.

### 49.5.1 Silent mode

The bxCAN can be put in Silent mode by setting the SILM bit in the CAN\_BTR register.

In Silent mode, the bxCAN is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the bxCAN has to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the CAN Core monitors this dominant bit, although the CAN bus may remain in recessive state. Silent mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames).

Figure 567. bxCAN in silent mode

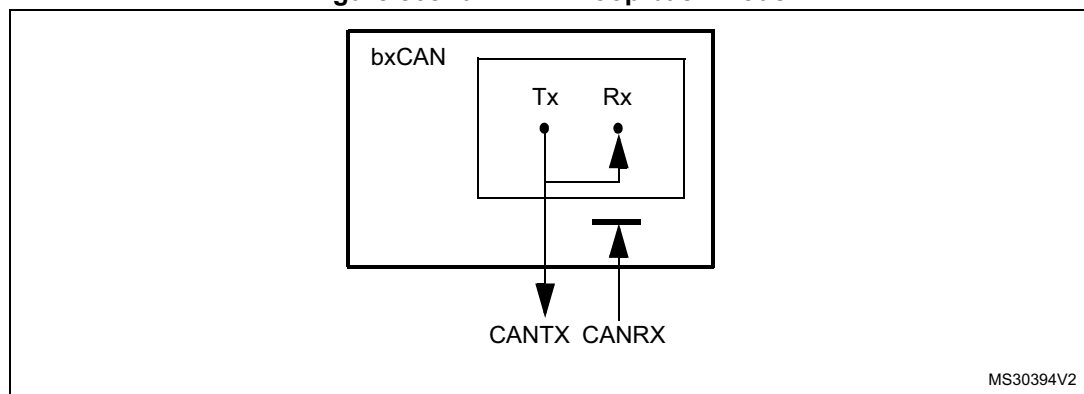


MS30393V2

### 49.5.2 Loop back mode

The bxCAN can be set in Loop Back Mode by setting the LBKM bit in the CAN\_BTR register. In Loop Back Mode, the bxCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) in a Receive mailbox.

Figure 568. bxCAN in loop back mode



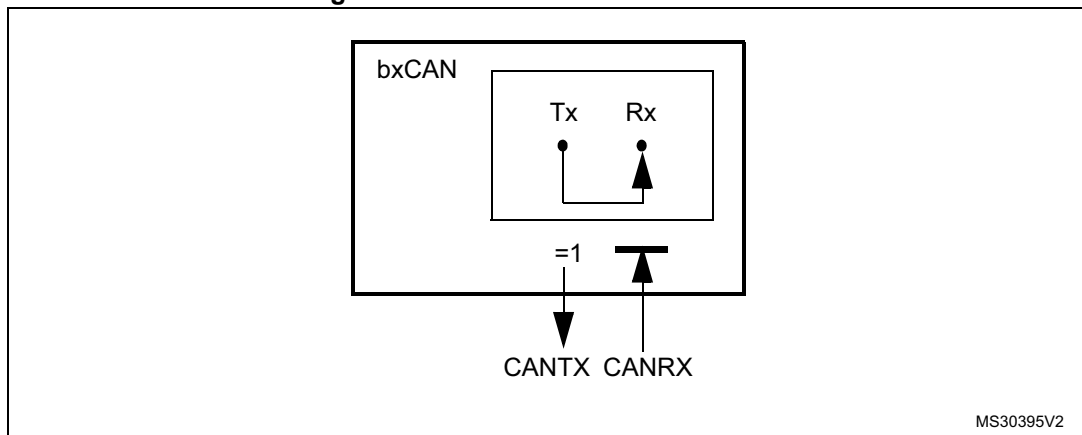
MS30394V2

This mode is provided for self-test functions. To be independent of external events, the CAN Core ignores acknowledge errors (no dominant bit sampled in the acknowledge slot of a data / remote frame) in Loop Back Mode. In this mode, the bxCAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CANRX input pin is disregarded by the bxCAN. The transmitted messages can be monitored on the CANTX pin.

### 49.5.3 Loop back combined with silent mode

It is also possible to combine Loop Back mode and Silent mode by setting the LBKM and SILM bits in the CAN\_BTR register. This mode can be used for a “Hot Selftest”, meaning the bxCAN can be tested like in Loop Back mode but without affecting a running CAN system connected to the CANTX and CANRX pins. In this mode, the CANRX pin is disconnected from the bxCAN and the CANTX pin is held recessive.

Figure 569. bxCAN in combined mode



## 49.6 Behavior in debug mode

When the microcontroller enters the debug mode (Cortex<sup>®</sup>-M4 core halted), the bxCAN continues to work normally or stops, depending on:

- the DBG\_CAN1\_STOP bit for CAN1 or the DBG\_CAN2\_STOP bit for CAN2 in the DBG module for the dual mode.
- the DBF bit in CAN\_MCR. For more details, refer to [Section 49.9.2: CAN control and status registers](#).

## 49.7 bxCAN functional description

### 49.7.1 Transmission handling

In order to transmit a message, the application must select one **empty** transmit mailbox, set up the identifier, the data length code (DLC) and the data before requesting the transmission by setting the corresponding TXRQ bit in the CAN\_TiRxR register. Once the mailbox has left **empty** state, the software no longer has write access to the mailbox registers. Immediately after the TXRQ bit has been set, the mailbox enters **pending** state and waits to become the highest priority mailbox, see *Transmit Priority*. As soon as the mailbox has the highest priority it will be **scheduled** for transmission. The transmission of the message of the

scheduled mailbox will start (enter **transmit** state) when the CAN bus becomes idle. Once the mailbox has been successfully transmitted, it will become **empty** again. The hardware indicates a successful transmission by setting the RQCP and TXOK bits in the CAN\_TSR register.

If the transmission fails, the cause is indicated by the ALST bit in the CAN\_TSR register in case of an Arbitration Lost, and/or the TERR bit, in case of transmission error detection.

### Transmit priority

By identifier

When more than one transmit mailbox is pending, the transmission order is given by the identifier of the message stored in the mailbox. The message with the lowest identifier value has the highest priority according to the arbitration of the CAN protocol. If the identifier values are equal, the lower mailbox number will be scheduled first.

By transmit request order

The transmit mailboxes can be configured as a transmit FIFO by setting the TXFP bit in the CAN\_MCR register. In this mode the priority order is given by the transmit request order.

This mode is very useful for segmented transmission.

### Abort

A transmission request can be aborted by the user setting the ABRQ bit in the CAN\_TSR register. In **pending** or **scheduled** state, the mailbox is aborted immediately. An abort request while the mailbox is in **transmit** state can have two results. If the mailbox is transmitted successfully the mailbox becomes **empty** with the TXOK bit set in the CAN\_TSR register. If the transmission fails, the mailbox becomes **scheduled**, the transmission is aborted and becomes **empty** with TXOK cleared. In all cases the mailbox will become **empty** again at least at the end of the current transmission.

### Non automatic retransmission mode

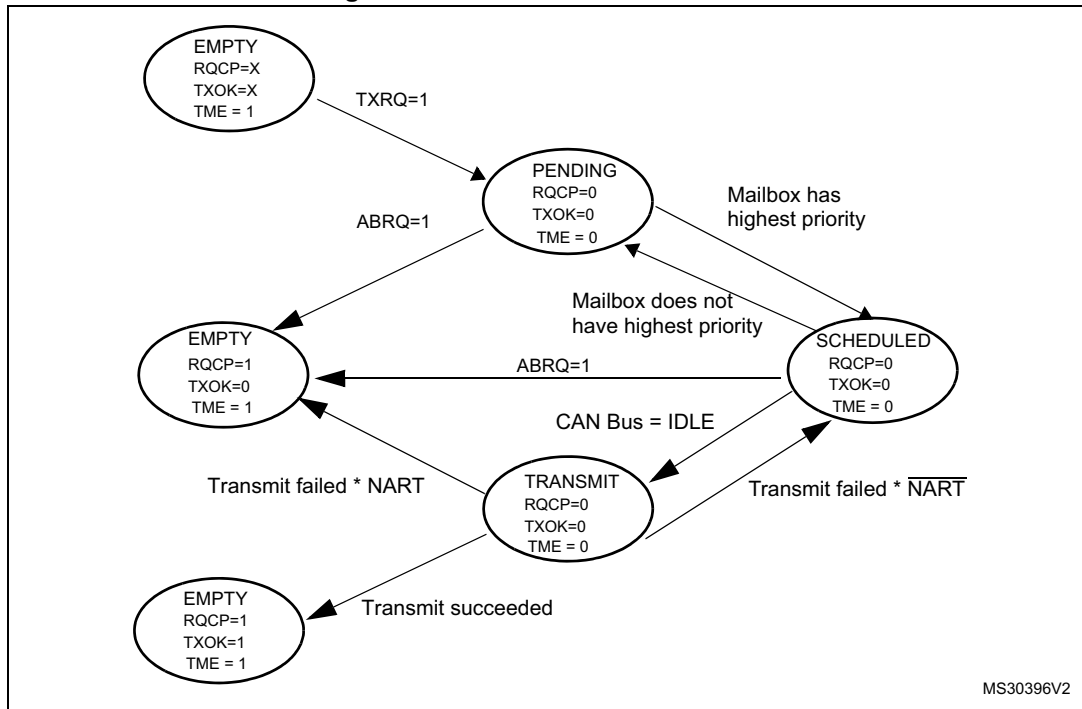
This mode has been implemented in order to fulfill the requirement of the Time Triggered Communication option of the CAN standard. To configure the hardware in this mode the NART bit in the CAN\_MCR register must be set.

In this mode, each transmission is started only once. If the first attempt fails, due to an arbitration loss or an error, the hardware will not automatically restart the message transmission.

At the end of the first transmission attempt, the hardware considers the request as completed and sets the RQCP bit in the CAN\_TSR register. The result of the transmission is indicated in the CAN\_TSR register by the TXOK, ALST and TERR bits.



Figure 570. Transmit mailbox states



MS30396V2

### 49.7.2 Time triggered communication mode

In this mode, the internal counter of the CAN hardware is activated and used to generate the Time Stamp value stored in the CAN\_RDTxR/CAN\_TDTxR registers, respectively (for Rx and Tx mailboxes). The internal counter is incremented each CAN bit time (refer to [Section 49.7.7: Bit timing](#)). The internal counter is captured on the sample point of the Start Of Frame bit in both reception and transmission.

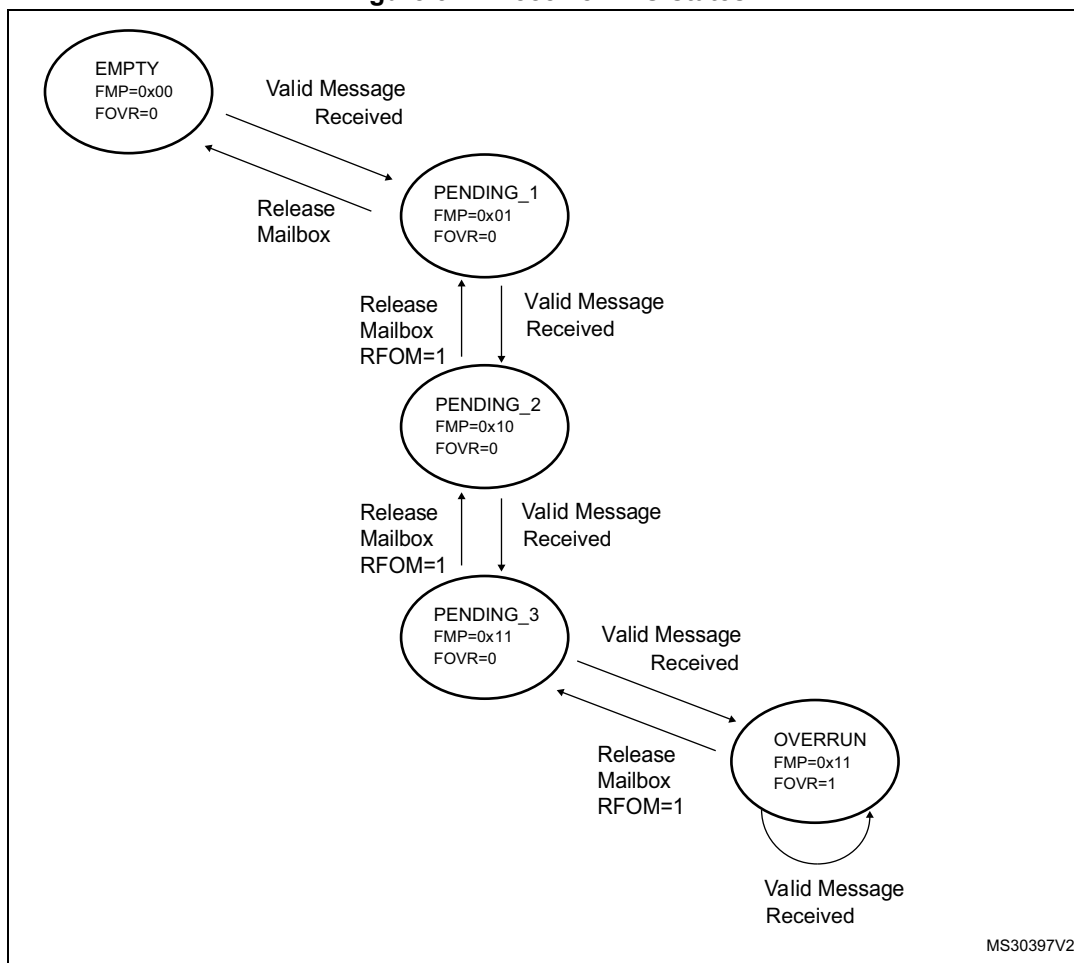
### 49.7.3 Reception handling

For the reception of CAN messages, three mailboxes organized as a FIFO are provided. In order to save CPU load, simplify the software and guarantee data consistency, the FIFO is managed completely by hardware. The application accesses the messages stored in the FIFO through the FIFO output mailbox.

#### Valid message

A received message is considered as valid **when** it has been received correctly according to the CAN protocol (no error until the last but one bit of the EOF field) **and** It passed through the identifier filtering successfully, see [Section 49.7.4: Identifier filtering](#).

Figure 571. Receive FIFO states



**FIFO management**

Starting from the **empty** state, the first valid message received is stored in the FIFO which becomes **pending\_1**. The hardware signals the event setting the FMP[1:0] bits in the CAN\_RFR register to the value 01b. The message is available in the FIFO output mailbox. The software reads out the mailbox content and releases it by setting the RFOM bit in the CAN\_RFR register. The FIFO becomes **empty** again. If a new valid message has been received in the meantime, the FIFO stays in **pending\_1** state and the new message is available in the output mailbox.

If the application does not release the mailbox, the next valid message will be stored in the FIFO which enters **pending\_2** state (FMP[1:0] = 10b). The storage process is repeated for the next valid message putting the FIFO into **pending\_3** state (FMP[1:0] = 11b). At this point, the software must release the output mailbox by setting the RFOM bit, so that a mailbox is free to store the next valid message. Otherwise the next valid message received will cause a loss of message.

Refer also to [Section 49.7.5: Message storage](#)

## Overrun

Once the FIFO is in **pending\_3** state (i.e. the three mailboxes are full) the next valid message reception will lead to an **overrun** and a message will be lost. The hardware signals the overrun condition by setting the FOVR bit in the CAN\_RFR register. Which message is lost depends on the configuration of the FIFO:

- If the FIFO lock function is disabled (RFLM bit in the CAN\_MCR register cleared) the last message stored in the FIFO will be overwritten by the new incoming message. In this case the latest messages will be always available to the application.
- If the FIFO lock function is enabled (RFLM bit in the CAN\_MCR register set) the most recent message will be discarded and the software will have the three oldest messages in the FIFO available.

## Reception related interrupts

Once a message has been stored in the FIFO, the FMP[1:0] bits are updated and an interrupt request is generated if the FMPIE bit in the CAN\_IER register is set.

When the FIFO becomes full (i.e. a third message is stored) the FULL bit in the CAN\_RFR register is set and an interrupt is generated if the FFIE bit in the CAN\_IER register is set.

On overrun condition, the FOVR bit is set and an interrupt is generated if the FOVIE bit in the CAN\_IER register is set.

### 49.7.4 Identifier filtering

In the CAN protocol the identifier of a message is not associated with the address of a node but related to the content of the message. Consequently a transmitter broadcasts its message to all receivers. On message reception a receiver node decides - depending on the identifier value - whether the software needs the message or not. If the message is needed, it is copied into the SRAM. If not, the message must be discarded without intervention by the software.

To fulfill this requirement the bxCAN Controller provides 28 configurable and scalable filter banks (27-0) to the application, in order to receive only the messages the software needs.

This hardware filtering saves CPU resources which would be otherwise needed to perform filtering by software. Each filter bank x consists of two 32-bit registers, CAN\_FxR0 and CAN\_FxR1.

#### Scalable width

To optimize and adapt the filters to the application needs, each filter bank can be scaled independently. Depending on the filter scale a filter bank provides:

- One 32-bit filter for the STDID[10:0], EXTID[17:0], IDE and RTR bits.
- Two 16-bit filters for the STDID[10:0], RTR, IDE and EXTID[17:15] bits.

Refer to [Figure 572](#).

Furthermore, the filters can be configured in mask mode or in identifier list mode.

#### Mask mode

In **mask** mode the identifier registers are associated with mask registers specifying which bits of the identifier are handled as “must match” or as “don't care”.

### Identifier list mode

In **identifier list** mode, the mask registers are used as identifier registers. Thus instead of defining an identifier and a mask, two identifiers are specified, doubling the number of single identifiers. All bits of the incoming identifier must match the bits specified in the filter registers.

### Filter bank scale and mode configuration

The filter banks are configured by means of the corresponding CAN\_FMR register. To configure a filter bank it must be deactivated by clearing the FACT bit in the CAN\_FAR register. The filter scale is configured by means of the corresponding FSCx bit in the CAN\_FS1R register, refer to [Figure 572](#). The **identifier list** or **identifier mask** mode for the corresponding Mask/Identifier registers is configured by means of the FBMx bits in the CAN\_FMR register.

To filter a group of identifiers, configure the Mask/Identifier registers in mask mode.

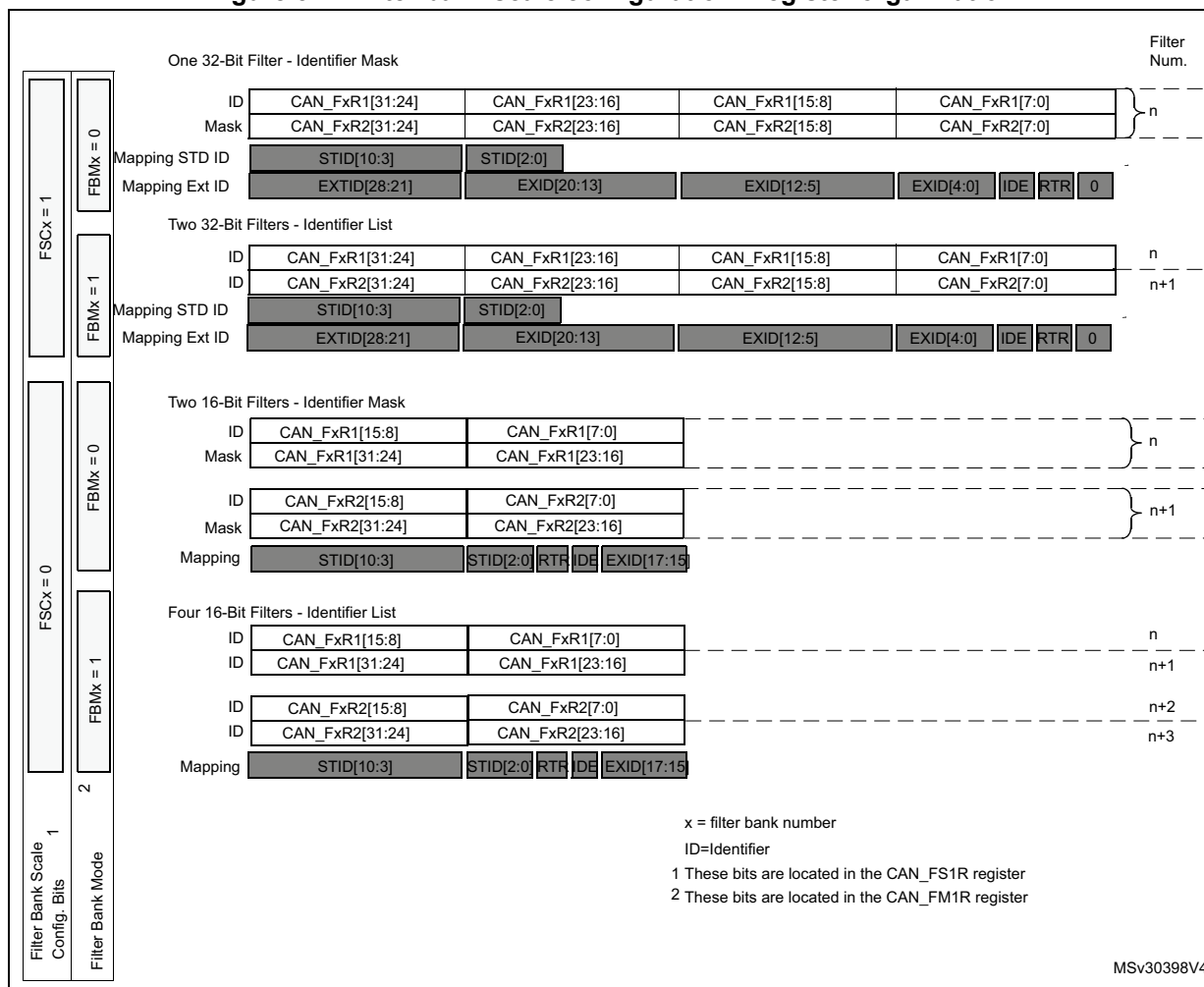
To select single identifiers, configure the Mask/Identifier registers in identifier list mode.

Filters not used by the application should be left deactivated.

Each filter within a filter bank is numbered (called the *Filter Number*) from 0 to a maximum dependent on the mode and the scale of each of the filter banks.

Concerning the filter configuration, refer to [Figure 572](#).

Figure 572. Filter bank scale configuration - register organization



### Filter match index

Once a message has been received in the FIFO it is available to the application. Typically, application data is copied into SRAM locations. To copy the data to the right location the application has to identify the data by means of the identifier. To avoid this, and to ease the access to the SRAM locations, the CAN controller provides a Filter Match Index.

This index is stored in the mailbox together with the message according to the filter priority rules. Thus each received message has its associated filter match index.

The Filter Match index can be used in two ways:

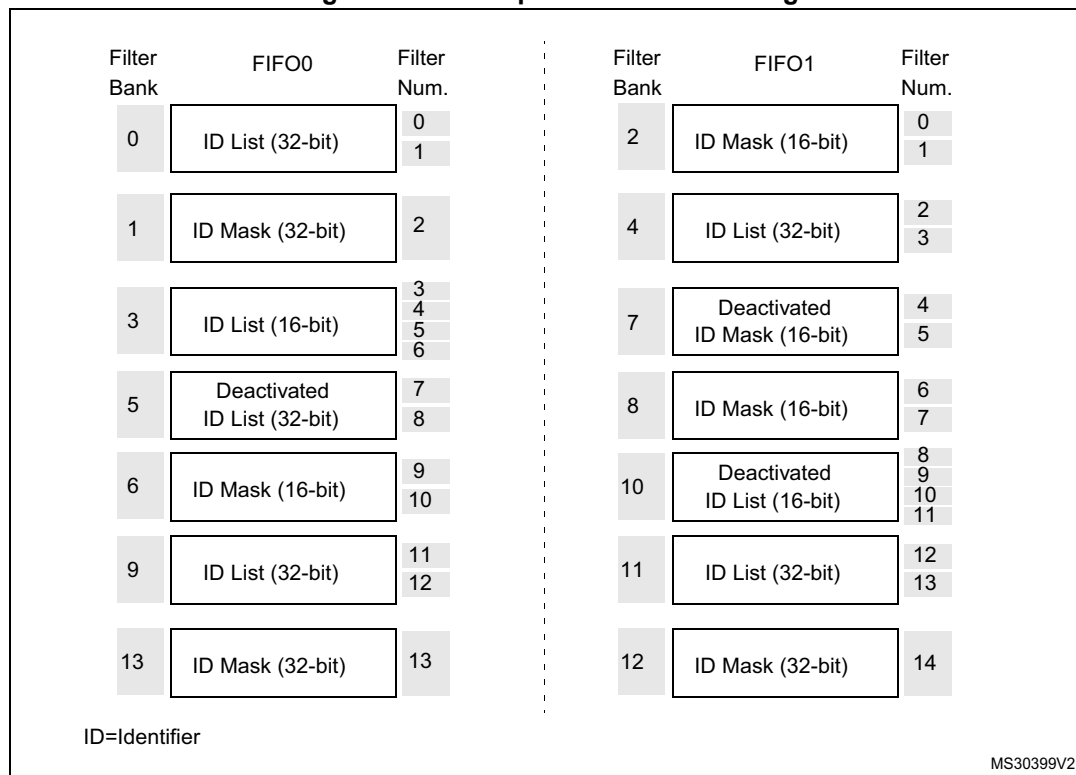
- Compare the Filter Match index with a list of expected values.
- Use the Filter Match Index as an index on an array to access the data destination location.

For non masked filters, the software no longer has to compare the identifier.

If the filter is masked the software reduces the comparison to the masked bits only.

The index value of the filter number does not take into account the activation state of the filter banks. In addition, two independent numbering schemes are used, one for each FIFO. Refer to [Figure 573](#) for an example.

**Figure 573. Example of filter numbering**

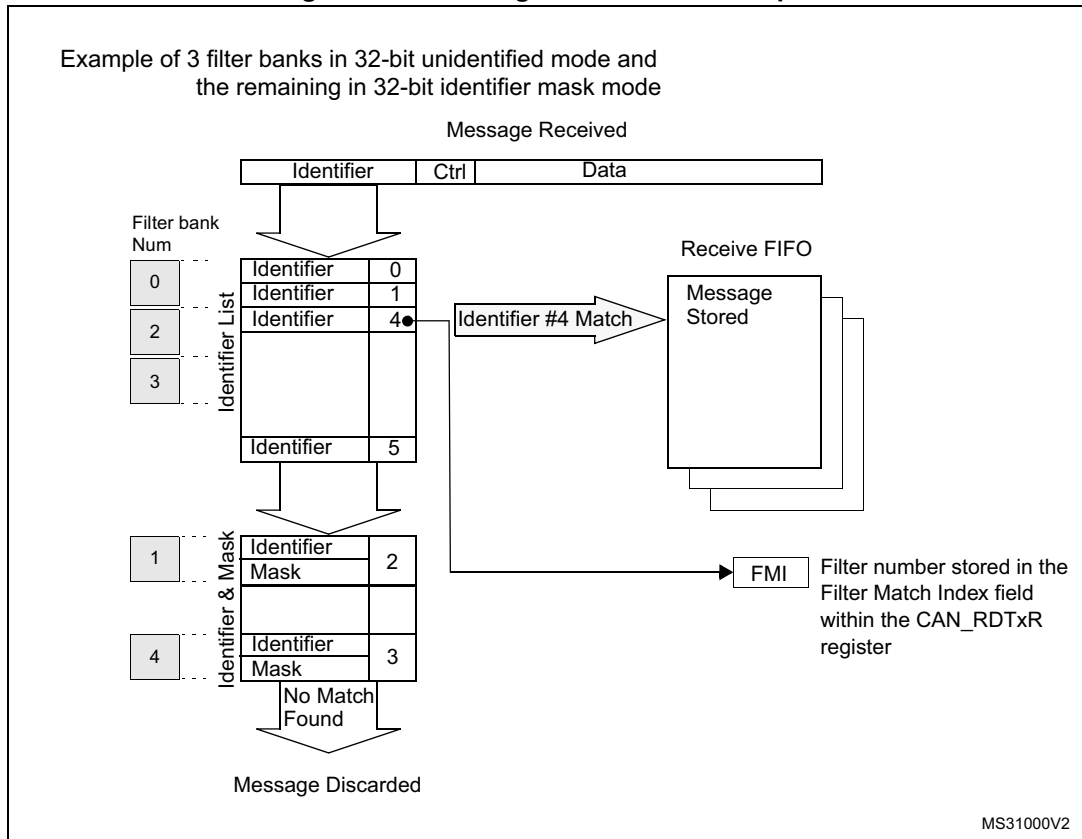


**Filter priority rules**

Depending on the filter combination it may occur that an identifier passes successfully through several filters. In this case the filter match value stored in the receive mailbox is chosen according to the following priority rules:

- A 32-bit filter takes priority over a 16-bit filter.
- For filters of equal scale, priority is given to the Identifier List mode over the Identifier Mask mode
- For filters of equal scale and mode, priority is given by the filter number (the lower the number, the higher the priority).

Figure 574. Filtering mechanism - example



The example above shows the filtering principle of the bxCAN. On reception of a message, the identifier is compared first with the filters configured in identifier list mode. If there is a match, the message is stored in the associated FIFO and the index of the matching filter is stored in the Filter Match Index. As shown in the example, the identifier matches with Identifier #2 thus the message content and FMI 2 is stored in the FIFO.

If there is no match, the incoming identifier is then compared with the filters configured in mask mode.

If the identifier does not match any of the identifiers configured in the filters, the message is discarded by hardware without disturbing the software.

### 49.7.5 Message storage

The interface between the software and the hardware for the CAN messages is implemented by means of mailboxes. A mailbox contains all information related to a message; identifier, data, control, status and time stamp information.

#### Transmit mailbox

The software sets up the message to be transmitted in an empty transmit mailbox. The status of the transmission is indicated by hardware in the CAN\_TSR register.

**Table 314. Transmit mailbox mapping**

Offset to transmit mailbox base address	Register name
0	CAN_TlRxR
4	CAN_TDTxR
8	CAN_TDLxR
12	CAN_TDHxR

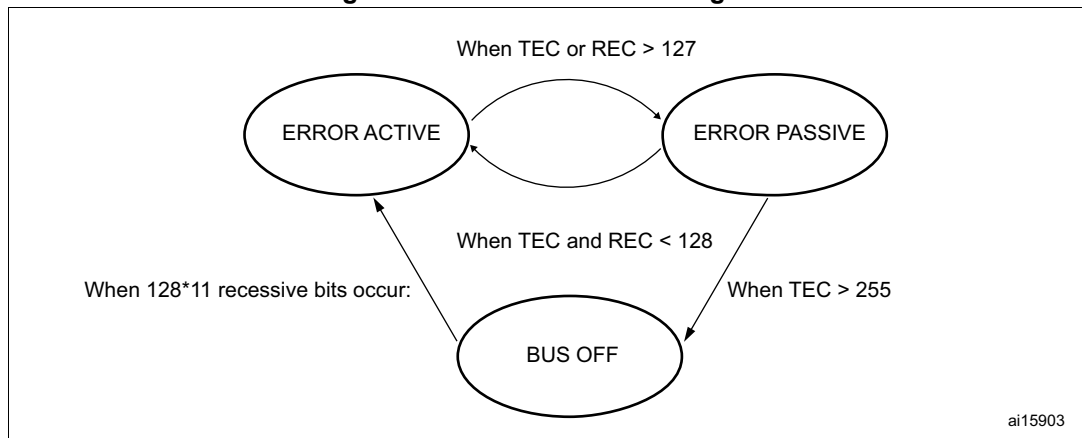
**Receive mailbox**

When a message has been received, it is available to the software in the FIFO output mailbox. Once the software has handled the message (e.g. read it) the software must release the FIFO output mailbox by means of the RFOM bit in the CAN\_RFR register to make the next incoming message available. The filter match index is stored in the MFMI field of the CAN\_RDTxR register. The 16-bit time stamp value is stored in the TIME[15:0] field of CAN\_RDTxR.

**Table 315. Receive mailbox mapping**

Offset to receive mailbox base address (bytes)	Register name
0	CAN_RlRxR
4	CAN_RDTxR
8	CAN_RDLxR
12	CAN_RDHxR

**Figure 575. CAN error state diagram**





### 49.7.6 Error management

The error management as described in the CAN protocol is handled entirely by hardware using a Transmit Error Counter (TEC value, in CAN\_ESR register) and a Receive Error Counter (REC value, in the CAN\_ESR register), which get incremented or decremented according to the error condition. For detailed information about TEC and REC management, refer to the CAN standard.

Both of them may be read by software to determine the stability of the network. Furthermore, the CAN hardware provides detailed information on the current error status in CAN\_ESR register. By means of the CAN\_IER register (ERRIE bit, etc.), the software can configure the interrupt generation on error detection in a very flexible way.

#### Bus-Off recovery

The Bus-Off state is reached when TEC is greater than 255, this state is indicated by BOFF bit in CAN\_ESR register. In Bus-Off state, the bxCAN is no longer able to transmit and receive messages.

Depending on the ABOM bit in the CAN\_MCR register bxCAN will recover from Bus-Off (become error active again) either automatically or on software request. But in both cases the bxCAN has to wait at least for the recovery sequence specified in the CAN standard (128 occurrences of 11 consecutive recessive bits monitored on CANRX).

If ABOM is set, the bxCAN will start the recovering sequence automatically after it has entered Bus-Off state.

If ABOM is cleared, the software must initiate the recovering sequence by requesting bxCAN to enter and to leave initialization mode.

*Note:* In initialization mode, bxCAN does not monitor the CANRX signal, therefore it cannot complete the recovery sequence. **To recover, bxCAN must be in normal mode.**

### 49.7.7 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

Its operation may be explained simply by splitting nominal bit time into three segments as follows:

- **Synchronization segment (SYNC\_SEG):** a bit change is expected to occur within this time segment. It has a fixed length of one time quantum ( $1 \times t_q$ ).
- **Bit segment 1 (BS1):** defines the location of the sample point. It includes the PROP\_SEG and PHASE\_SEG1 of the CAN standard. Its duration is programmable between 1 and 16 time quanta but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of the various nodes of the network.
- **Bit segment 2 (BS2):** defines the location of the transmit point. It represents the PHASE\_SEG2 of the CAN standard. Its duration is programmable between 1 and 8 time quanta but may also be automatically shortened to compensate for negative phase drifts.

The resynchronization Jump Width (SJW) defines an upper bound to the amount of lengthening or shortening of the bit segments. It is programmable between 1 and 4 time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC\_SEG, BS1 is extended by up to SJW so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC\_SEG, BS2 is shortened by up to SJW so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the Bit Timing Register (CAN\_BTR) is only possible while the device is in Standby mode.

*Note: For a detailed description of the CAN bit timing and resynchronization mechanism, refer to the ISO 11898 standard.*

**Figure 576. Bit timing**

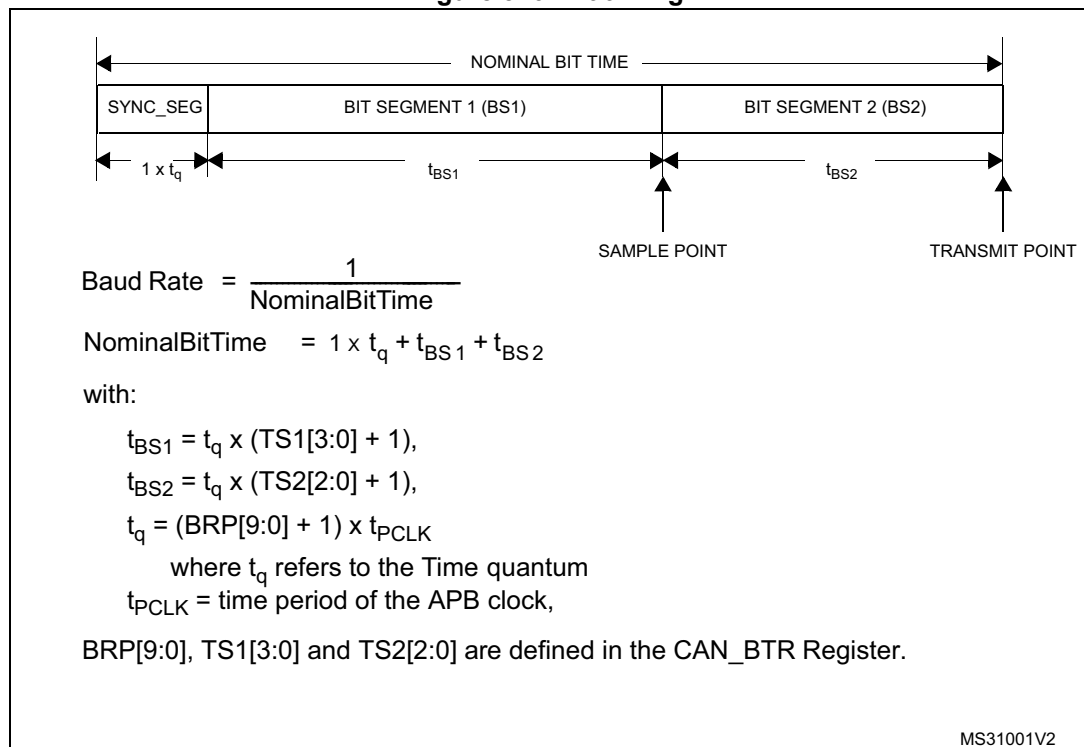
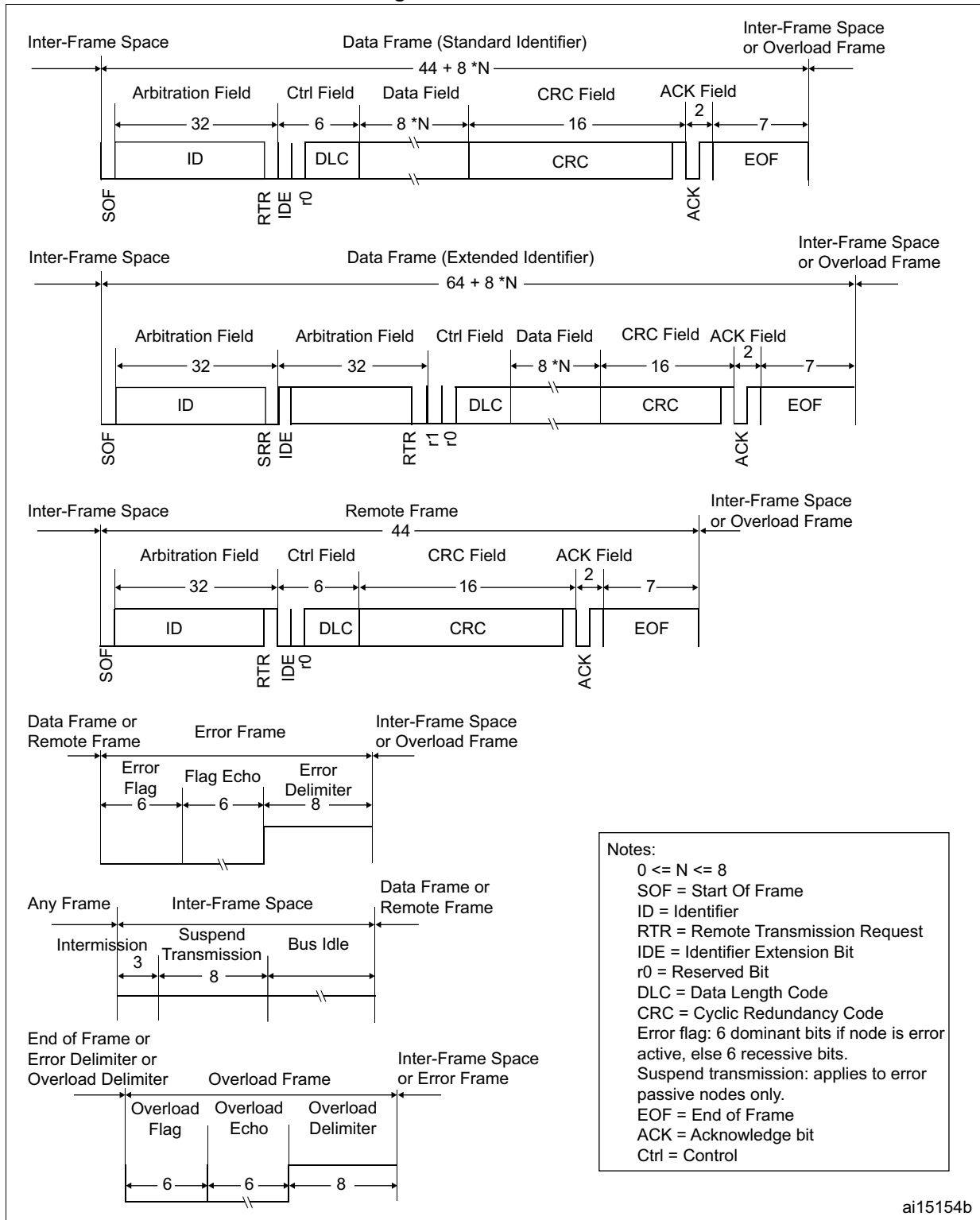


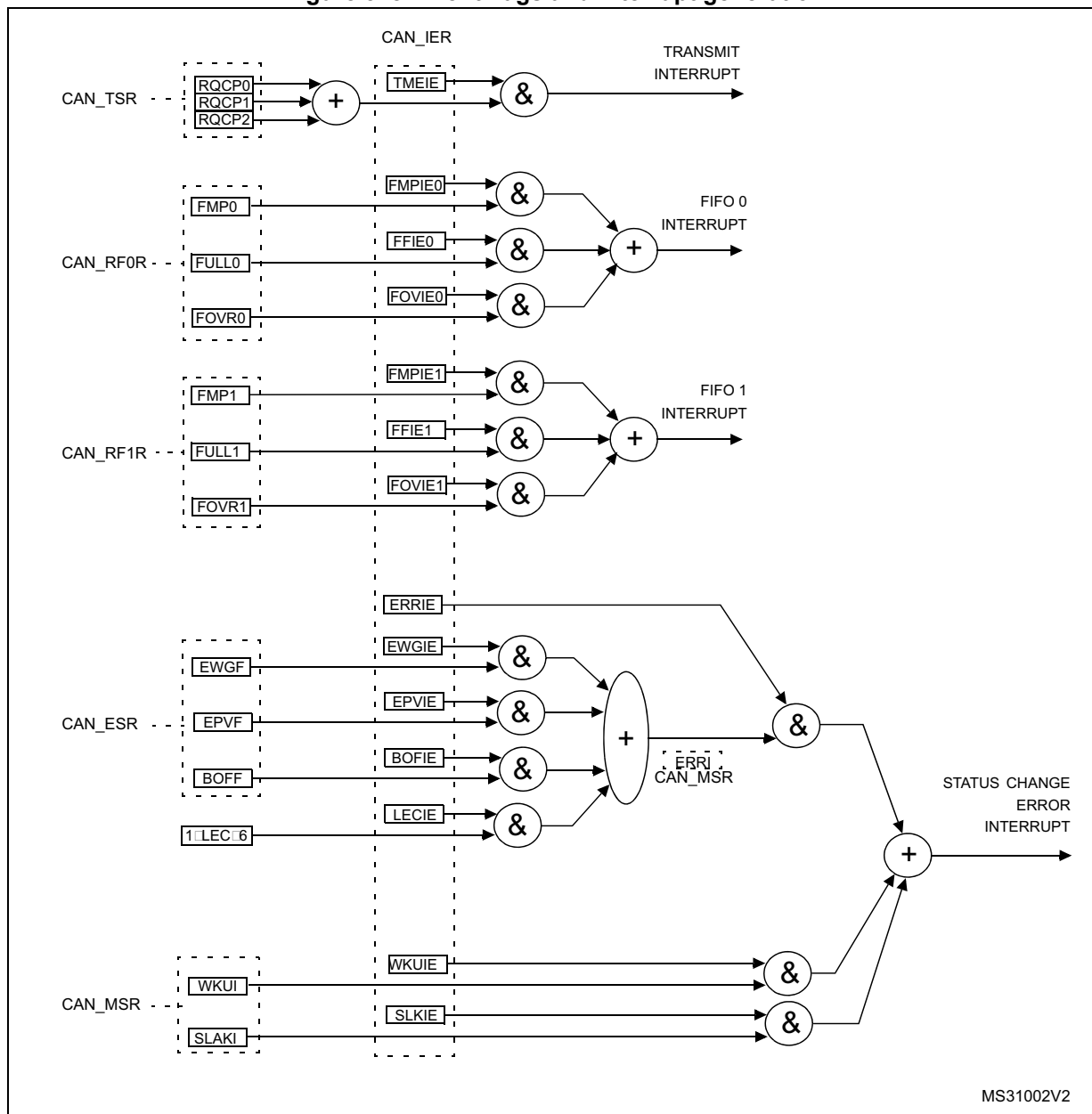
Figure 577. CAN frames



### 49.8 bxCAN interrupts

Four interrupt vectors are dedicated to bxCAN. Each interrupt source can be independently enabled or disabled by means of the CAN Interrupt Enable Register (CAN\_IER).

Figure 578. Event flags and interrupt generation



- The **transmit interrupt** can be generated by the following events:
  - Transmit mailbox 0 becomes empty, RQCP0 bit in the CAN\_TSR register set.
  - Transmit mailbox 1 becomes empty, RQCP1 bit in the CAN\_TSR register set.
  - Transmit mailbox 2 becomes empty, RQCP2 bit in the CAN\_TSR register set.
- The **FIFO 0 interrupt** can be generated by the following events:
  - Reception of a new message, FMP0 bits in the CAN\_RF0R register are not '00'.
  - FIFO0 full condition, FULL0 bit in the CAN\_RF0R register set.
  - FIFO0 overrun condition, FOVR0 bit in the CAN\_RF0R register set.
- The **FIFO 1 interrupt** can be generated by the following events:
  - Reception of a new message, FMP1 bits in the CAN\_RF1R register are not '00'.
  - FIFO1 full condition, FULL1 bit in the CAN\_RF1R register set.
  - FIFO1 overrun condition, FOVR1 bit in the CAN\_RF1R register set.
- The **error and status change interrupt** can be generated by the following events:
  - Error condition, for more details on error conditions refer to the CAN Error Status register (CAN\_ESR).
  - Wakeup condition, SOF monitored on the CAN Rx signal.
  - Entry into Sleep mode.

## 49.9 CAN registers

The peripheral registers have to be accessed by words (32 bits).

### 49.9.1 Register access protection

Erroneous access to certain configuration registers can cause the hardware to temporarily disturb the whole CAN network. Therefore the CAN\_BTR register can be modified by software only while the CAN hardware is in initialization mode.

Although the transmission of incorrect data will not cause problems at the CAN network level, it can severely disturb the application. A transmit mailbox can be only modified by software while it is in empty state, refer to [Figure 570: Transmit mailbox states](#).

The filter values can be modified either deactivating the associated filter banks or by setting the FINIT bit. Moreover, the modification of the filter configuration (scale, mode and FIFO assignment) in CAN\_FMR, CAN\_FSxR and CAN\_FFAR registers can only be done when the filter initialization mode is set (FINIT=1) in the CAN\_FMR register.

### 49.9.2 CAN control and status registers

Refer to [Section 1.2](#) for a list of abbreviations used in register descriptions.

#### CAN master control register (CAN\_MCR)

Address offset: 0x00

Reset value: 0x0001 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBF
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
rs								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DBF**: Debug freeze

0: CAN working during debug

1: CAN reception/transmission frozen during debug. Reception FIFOs can still be accessed/controlled normally.

Bit 15 **RESET**: bxCAN software master reset

0: Normal operation.

1: Force a master reset of the bxCAN -> Sleep mode activated after reset (FMP bits and CAN\_MCR register are initialized to the reset values). This bit is automatically reset to 0.

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **TTCM**: Time triggered communication mode

0: Time Triggered Communication mode disabled.

1: Time Triggered Communication mode enabled

*Note: For more information on Time Triggered Communication mode, refer to [Section 49.7.2: Time triggered communication mode](#).*

Bit 6 **ABOM**: Automatic bus-off management

This bit controls the behavior of the CAN hardware on leaving the Bus-Off state.

0: The Bus-Off state is left on software request, once 128 occurrences of 11 recessive bits have been monitored and the software has first set and cleared the INRQ bit of the CAN\_MCR register.

1: The Bus-Off state is left automatically by hardware once 128 occurrences of 11 recessive bits have been monitored.

For detailed information on the Bus-Off state refer to [Section 49.7.6: Error management](#).

Bit 5 **AWUM**: Automatic wakeup mode

This bit controls the behavior of the CAN hardware on message reception during Sleep mode.

0: The Sleep mode is left on software request by clearing the SLEEP bit of the CAN\_MCR register.

1: The Sleep mode is left automatically by hardware on CAN message detection.

The SLEEP bit of the CAN\_MCR register and the SLAK bit of the CAN\_MSR register are cleared by hardware.

Bit 4 **NART**: No automatic retransmission

0: The CAN hardware will automatically retransmit the message until it has been successfully transmitted according to the CAN standard.

1: A message will be transmitted only once, independently of the transmission result (successful, error or arbitration lost).

- Bit 3 **RFLM**: Receive FIFO locked mode
  - 0: Receive FIFO not locked on overrun. Once a receive FIFO is full the next incoming message will overwrite the previous one.
  - 1: Receive FIFO locked against overrun. Once a receive FIFO is full the next incoming message will be discarded.
  
- Bit 2 **TXFP**: Transmit FIFO priority
  - This bit controls the transmission order when several mailboxes are pending at the same time.
  - 0: Priority driven by the identifier of the message
  - 1: Priority driven by the request order (chronologically)
  
- Bit 1 **SLEEP**: Sleep mode request
  - This bit is set by software to request the CAN hardware to enter the Sleep mode. Sleep mode will be entered as soon as the current CAN activity (transmission or reception of a CAN frame) has been completed.
  - This bit is cleared by software to exit Sleep mode.
  - This bit is cleared by hardware when the AWUM bit is set and a SOF bit is detected on the CAN Rx signal.
  - This bit is set after reset - CAN starts in Sleep mode.
  
- Bit 0 **INRQ**: Initialization request
  - The software clears this bit to switch the hardware into normal mode. Once 11 consecutive recessive bits have been monitored on the Rx signal the CAN hardware is synchronized and ready for transmission and reception. Hardware signals this event by clearing the INAK bit in the CAN\_MSR register.
  - Software sets this bit to request the CAN hardware to enter initialization mode. Once software has set the INRQ bit, the CAN hardware waits until the current CAN activity (transmission or reception) is completed before entering the initialization mode. Hardware signals this event by setting the INAK bit in the CAN\_MSR register.

**CAN master status register (CAN\_MSR)**

Address offset: 0x04  
 Reset value: 0x0000 0C02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RX	SAMP	RXM	TXM	Res.	Res.	Res.	SLAKI	WKUI	ERRI	SLAK	INAK
				r	r	r	r				rc_w1	rc_w1	rc_w1	r	r

- Bits 31:12 Reserved, must be kept at reset value.
- Bit 11 **RX**: CAN Rx signal
  - Monitors the actual value of the **CAN\_RX** Pin.
- Bit 10 **SAMP**: Last sample point
  - The value of RX on the last sample point (current received bit value).
- Bit 9 **RXM**: Receive mode
  - The CAN hardware is currently receiver.



Bit 8 **TXM**: Transmit mode  
 The CAN hardware is currently transmitter.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SLAKI**: Sleep acknowledge interrupt  
 When SLKIE=1, this bit is set by hardware to signal that the bxCAN has entered Sleep Mode. When set, this bit generates a status change interrupt if the SLKIE bit in the CAN\_IER register is set.  
 This bit is cleared by software or by hardware, when SLAK is cleared.  
*Note: When SLKIE=0, no polling on SLAKI is possible. In this case the SLAK bit can be polled.*

Bit 3 **WKUI**: Wakeup interrupt  
 This bit is set by hardware to signal that a SOF bit has been detected while the CAN hardware was in Sleep mode. Setting this bit generates a status change interrupt if the WKUIE bit in the CAN\_IER register is set.  
 This bit is cleared by software.

Bit 2 **ERRI**: Error interrupt  
 This bit is set by hardware when a bit of the CAN\_ESR has been set on error detection and the corresponding interrupt in the CAN\_IER is enabled. Setting this bit generates a status change interrupt if the ERRIE bit in the CAN\_IER register is set.  
 This bit is cleared by software.

Bit 1 **SLAK**: Sleep acknowledge  
 This bit is set by hardware and indicates to the software that the CAN hardware is now in Sleep mode. This bit acknowledges the Sleep mode request from the software (set SLEEP bit in CAN\_MCR register).  
 This bit is cleared by hardware when the CAN hardware has left Sleep mode (to be synchronized on the CAN bus). To be synchronized the hardware has to monitor a sequence of 11 consecutive recessive bits on the CAN RX signal.  
*Note: The process of leaving Sleep mode is triggered when the SLEEP bit in the CAN\_MCR register is cleared. Refer to the AWUM bit of the CAN\_MCR register description for detailed information for clearing SLEEP bit*

Bit 0 **INAK**: Initialization acknowledge  
 This bit is set by hardware and indicates to the software that the CAN hardware is now in initialization mode. This bit acknowledges the initialization request from the software (set INRQ bit in CAN\_MCR register).  
 This bit is cleared by hardware when the CAN hardware has left the initialization mode (to be synchronized on the CAN bus). To be synchronized the hardware has to monitor a sequence of 11 consecutive recessive bits on the CAN RX signal.

**CAN transmit status register (CAN\_TSR)**

Address offset: 0x08  
 Reset value: 0x1C00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOW2	LOW1	LOW0	TME2	TME1	TME0	CODE[1:0]		ABRQ2	Res.	Res.	Res.	TERR2	ALST2	TXOK2	RQCP2
r	r	r	r	r	r	r	r	rs				rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRQ1	Res.	Res.	Res.	TERR1	ALST1	TXOK1	RQCP1	ABRQ0	Res.	Res.	Res.	TERR0	ALST0	TXOK0	RQCP0
rs				rc_w1	rc_w1	rc_w1	rc_w1	rs				rc_w1	rc_w1	rc_w1	rc_w1





- Bit 31 **LOW2**: Lowest priority flag for mailbox 2  
This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 2 has the lowest priority.
- Bit 30 **LOW1**: Lowest priority flag for mailbox 1  
This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 1 has the lowest priority.
- Bit 29 **LOW0**: Lowest priority flag for mailbox 0  
This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 0 has the lowest priority.  
*Note: The LOW[2:0] bits are set to zero when only one mailbox is pending.*
- Bit 28 **TME2**: Transmit mailbox 2 empty  
This bit is set by hardware when no transmit request is pending for mailbox 2.
- Bit 27 **TME1**: Transmit mailbox 1 empty  
This bit is set by hardware when no transmit request is pending for mailbox 1.
- Bit 26 **TME0**: Transmit mailbox 0 empty  
This bit is set by hardware when no transmit request is pending for mailbox 0.
- Bits 25:24 **CODE[1:0]**: Mailbox code  
In case at least one transmit mailbox is free, the code value is equal to the number of the next transmit mailbox free.  
In case all transmit mailboxes are pending, the code value is equal to the number of the transmit mailbox with the lowest priority.
- Bit 23 **ABRQ2**: Abort request for mailbox 2  
Set by software to abort the transmission request for the corresponding mailbox.  
Cleared by hardware when the mailbox becomes empty.  
Setting this bit has no effect when the mailbox is not pending for transmission.
- Bits 22:20 Reserved, must be kept at reset value.
- Bit 19 **TERR2**: Transmission error of mailbox 2  
This bit is set when the previous TX failed due to an error.
- Bit 18 **ALST2**: Arbitration lost for mailbox 2  
This bit is set when the previous TX failed due to an arbitration lost.
- Bit 17 **TXOK2**: Transmission OK of mailbox 2  
The hardware updates this bit after each transmission attempt.  
0: The previous transmission failed  
1: The previous transmission was successful  
This bit is set by hardware when the transmission request on mailbox 2 has been completed successfully. Refer to [Figure 570](#).
- Bit 16 **RQCP2**: Request completed mailbox2  
Set by hardware when the last request (transmit or abort) has been performed.  
Cleared by software writing a "1" or by hardware on transmission request (TXRQ2 set in CAN\_TMD2R register).  
Clearing this bit clears all the status bits (TXOK2, ALST2 and TERR2) for Mailbox 2.
- Bit 15 **ABRQ1**: Abort request for mailbox 1  
Set by software to abort the transmission request for the corresponding mailbox.  
Cleared by hardware when the mailbox becomes empty.  
Setting this bit has no effect when the mailbox is not pending for transmission.
- Bits 14:12 Reserved, must be kept at reset value.

- Bit 11 **TERR1**: Transmission error of mailbox1  
This bit is set when the previous TX failed due to an error.
- Bit 10 **ALST1**: Arbitration lost for mailbox1  
This bit is set when the previous TX failed due to an arbitration lost.
- Bit 9 **TXOK1**: Transmission OK of mailbox1  
The hardware updates this bit after each transmission attempt.  
0: The previous transmission failed  
1: The previous transmission was successful  
This bit is set by hardware when the transmission request on mailbox 1 has been completed successfully. Refer to [Figure 570](#)
- Bit 8 **RQCP1**: Request completed mailbox1  
Set by hardware when the last request (transmit or abort) has been performed.  
Cleared by software writing a “1” or by hardware on transmission request (TXRQ1 set in CAN\_TI1R register).  
Clearing this bit clears all the status bits (TXOK1, ALST1 and TERR1) for Mailbox 1.
- Bit 7 **ABRQ0**: Abort request for mailbox0  
Set by software to abort the transmission request for the corresponding mailbox.  
Cleared by hardware when the mailbox becomes empty.  
Setting this bit has no effect when the mailbox is not pending for transmission.
- Bits 6:4 Reserved, must be kept at reset value.
- Bit 3 **TERR0**: Transmission error of mailbox0  
This bit is set when the previous TX failed due to an error.
- Bit 2 **ALST0**: Arbitration lost for mailbox0  
This bit is set when the previous TX failed due to an arbitration lost.
- Bit 1 **TXOK0**: Transmission OK of mailbox0  
The hardware updates this bit after each transmission attempt.  
0: The previous transmission failed  
1: The previous transmission was successful  
This bit is set by hardware when the transmission request on mailbox 1 has been completed successfully. Refer to [Figure 570](#)
- Bit 0 **RQCP0**: Request completed mailbox0  
Set by hardware when the last request (transmit or abort) has been performed.  
Cleared by software writing a “1” or by hardware on transmission request (TXRQ0 set in CAN\_TI0R register).  
Clearing this bit clears all the status bits (TXOK0, ALST0 and TERR0) for Mailbox 0.

**CAN receive FIFO 0 register (CAN\_RF0R)**

Address offset: 0x0C  
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOM0	FOVR0	FULL0	Res.	FMP0[1:0]	
										rs	rc_w1	rc_w1		r	r



Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **RFOM0**: Release FIFO 0 output mailbox

Set by software to release the output mailbox of the FIFO. The output mailbox can only be released when at least one message is pending in the FIFO. Setting this bit when the FIFO is empty has no effect. If at least two messages are pending in the FIFO, the software has to release the output mailbox to access the next message.  
Cleared by hardware when the output mailbox has been released.

Bit 4 **FOVR0**: FIFO 0 overrun

This bit is set by hardware when a new message has been received and passed the filter while the FIFO was full.  
This bit is cleared by software.

Bit 3 **FULL0**: FIFO 0 full

Set by hardware when three messages are stored in the FIFO.  
This bit is cleared by software.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **FMP0[1:0]**: FIFO 0 message pending

These bits indicate how many messages are pending in the receive FIFO.  
FMP is increased each time the hardware stores a new message in to the FIFO. FMP is decreased each time the software releases the output mailbox by setting the RFOM0 bit.

### CAN receive FIFO 1 register (CAN\_RF1R)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOM1	FOVR1	FULL1	Res.	FMP1[1:0]	
										rs	rc_w1	rc_w1		r	r

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **RFOM1**: Release FIFO 1 output mailbox

Set by software to release the output mailbox of the FIFO. The output mailbox can only be released when at least one message is pending in the FIFO. Setting this bit when the FIFO is empty has no effect. If at least two messages are pending in the FIFO, the software has to release the output mailbox to access the next message.  
Cleared by hardware when the output mailbox has been released.

Bit 4 **FOVR1**: FIFO 1 overrun

This bit is set by hardware when a new message has been received and passed the filter while the FIFO was full.  
This bit is cleared by software.

Bit 3 **FULL1**: FIFO 1 full

Set by hardware when three messages are stored in the FIFO.  
This bit is cleared by software.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **FMP1[1:0]**: FIFO 1 message pending

These bits indicate how many messages are pending in the receive FIFO1.  
FMP1 is increased each time the hardware stores a new message in to the FIFO1. FMP is decreased each time the software releases the output mailbox by setting the RFOM1 bit.

**CAN interrupt enable register (CAN\_IER)**

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLKIE	WKUIE
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIE	Res.	Res.	Res.	LEC IE	BOF IE	EPV IE	EWG IE	Res.	FOV IE1	FF IE1	FMP IE1	FOV IE0	FF IE0	FMP IE0	TME IE
rw				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **SLKIE**: Sleep interrupt enable

0: No interrupt when SLAKI bit is set.  
1: Interrupt generated when SLAKI bit is set.

Bit 16 **WKUIE**: Wakeup interrupt enable

0: No interrupt when WKUI is set.  
1: Interrupt generated when WKUI bit is set.

Bit 15 **ERRIE**: Error interrupt enable

0: No interrupt will be generated when an error condition is pending in the CAN\_ESR.  
1: An interrupt will be generation when an error condition is pending in the CAN\_ESR.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **LECIE**: Last error code interrupt enable

0: ERRI bit will not be set when the error code in LEC[2:0] is set by hardware on error detection.  
1: ERRI bit will be set when the error code in LEC[2:0] is set by hardware on error detection.

Bit 10 **BOFIE**: Bus-off interrupt enable

0: ERRI bit will not be set when BOFF is set.  
1: ERRI bit will be set when BOFF is set.

Bit 9 **EPVIE**: Error passive interrupt enable

0: ERRI bit will not be set when EPVF is set.  
1: ERRI bit will be set when EPVF is set.

- Bit 8 **EWGIE**: Error warning interrupt enable  
 0: ERRI bit will not be set when EWGF is set.  
 1: ERRI bit will be set when EWGF is set.
- Bit 7 Reserved, must be kept at reset value.
- Bit 6 **FOVIE1**: FIFO overrun interrupt enable  
 0: No interrupt when FOVR is set.  
 1: Interrupt generation when FOVR is set.
- Bit 5 **FFIE1**: FIFO full interrupt enable  
 0: No interrupt when FULL bit is set.  
 1: Interrupt generated when FULL bit is set.
- Bit 4 **FMPIE1**: FIFO message pending interrupt enable  
 0: No interrupt generated when state of FMP[1:0] bits are not 00b.  
 1: Interrupt generated when state of FMP[1:0] bits are not 00b.
- Bit 3 **FOVIE0**: FIFO overrun interrupt enable  
 0: No interrupt when FOVR bit is set.  
 1: Interrupt generated when FOVR bit is set.
- Bit 2 **FFIE0**: FIFO full interrupt enable  
 0: No interrupt when FULL bit is set.  
 1: Interrupt generated when FULL bit is set.
- Bit 1 **FMPIE0**: FIFO message pending interrupt enable  
 0: No interrupt generated when state of FMP[1:0] bits are not 00b.  
 1: Interrupt generated when state of FMP[1:0] bits are not 00b.
- Bit 0 **TMEIE**: Transmit mailbox empty interrupt enable  
 0: No interrupt when RQCPx bit is set.  
 1: Interrupt generated when RQCPx bit is set.  
*Note: Refer to [Section 49.8: bxCAN interrupts](#).*

**CAN error status register (CAN\_ESR)**

Address offset: 0x18  
 Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REC[7:0]								TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LEC[2:0]			Res.	BOFF	EPVF	EWGF
									r/w	r/w	r/w		r	r	r

Bits 31:24 **REC[7:0]**: Receive error counter

The implementing part of the fault confinement mechanism of the CAN protocol. In case of an error during reception, this counter is incremented by 1 or by 8 depending on the error condition as defined by the CAN standard. After every successful reception the counter is decremented by 1 or reset to 120 if its value was higher than 128. When the counter value exceeds 127, the CAN controller enters the error passive state.

Bits 23:16 **TEC[7:0]**: Least significant byte of the 9-bit transmit error counter

The implementing part of the fault confinement mechanism of the CAN protocol.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **LEC[2:0]**: Last error code

This field is set by hardware and holds a code which indicates the error condition of the last error detected on the CAN bus. If a message has been transferred (reception or transmission) without error, this field will be cleared to '0'.

The LEC[2:0] bits can be set to value 0b111 by software. They are updated by hardware to indicate the current communication status.

- 000: No Error
- 001: Stuff Error
- 010: Form Error
- 011: Acknowledgment Error
- 100: Bit recessive Error
- 101: Bit dominant Error
- 110: CRC Error
- 111: Set by software

Bit 3 Reserved, must be kept at reset value.

Bit 2 **BOFF**: Bus-off flag

This bit is set by hardware when it enters the bus-off state. The bus-off state is entered on TEC overflow, greater than 255, refer to [Section 49.7.6 on page 1835](#).

Bit 1 **EPVF**: Error passive flag

This bit is set by hardware when the Error Passive limit has been reached (Receive Error Counter or Transmit Error Counter > 127).

Bit 0 **EWGF**: Error warning flag

This bit is set by hardware when the warning limit has been reached (Receive Error Counter or Transmit Error Counter ≥ 96).

### CAN bit timing register (CAN\_BTR)

Address offset: 0x1C

Reset value: 0x0123 0000

This register can only be accessed by the software when the CAN hardware is in initialization mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
SILM	LBKM	Res.	Res.	Res.	Res.	SJW[1:0]		Res.	TS2[2:0]			TS1[3:0]				
rw	rw					rw	rw		rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	Res.	Res.	Res.	Res.	Res.	BRP[9:0]										
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

- Bit 31 **SILM**: Silent mode (debug)  
 0: Normal operation  
 1: Silent Mode
- Bit 30 **LBKM**: Loop back mode (debug)  
 0: Loop Back Mode disabled  
 1: Loop Back Mode enabled
- Bits 29:26 Reserved, must be kept at reset value.
- Bits 25:24 **SJW[1:0]**: Resynchronization jump width  
 These bits define the maximum number of time quanta the CAN hardware is allowed to lengthen or shorten a bit to perform the resynchronization.  
 $t_{RJW} = t_q \times (SJW[1:0] + 1)$
- Bit 23 Reserved, must be kept at reset value.
- Bits 22:20 **TS2[2:0]**: Time segment 2  
 These bits define the number of time quanta in Time Segment 2.  
 $t_{BS2} = t_q \times (TS2[2:0] + 1)$
- Bits 19:16 **TS1[3:0]**: Time segment 1  
 These bits define the number of time quanta in Time Segment 1  
 $t_{BS1} = t_q \times (TS1[3:0] + 1)$   
 For more information on bit timing, refer to [Section 49.7.7: Bit timing on page 1835](#).
- Bits 15:10 Reserved, must be kept at reset value.
- Bits 9:0 **BRP[9:0]**: Baud rate prescaler  
 These bits define the length of a time quanta.  
 $t_q = (BRP[9:0] + 1) \times t_{PCLK}$

### 49.9.3 CAN mailbox registers

This chapter describes the registers of the transmit and receive mailboxes. Refer to [Section 49.7.5: Message storage on page 1833](#) for detailed register mapping.

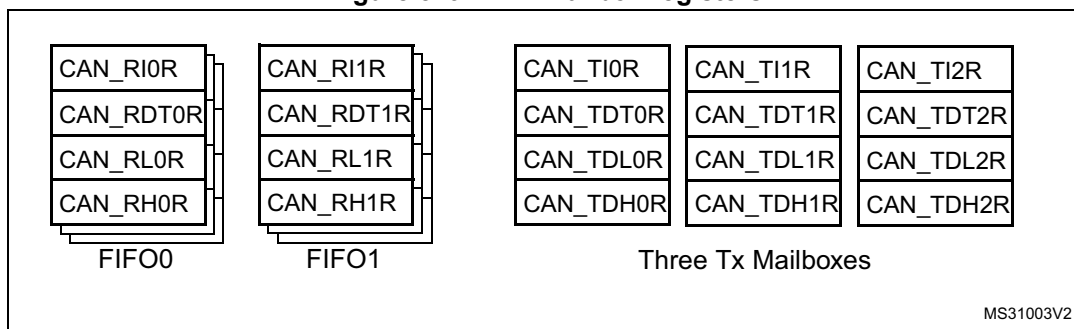
Transmit and receive mailboxes have the same registers except:

- The FMI field in the CAN\_RDTxR register.
- A receive mailbox is always write protected.
- A transmit mailbox is write-enabled only while empty, corresponding TME bit in the CAN\_TSR register set.

There are 3 TX Mailboxes and 2 RX Mailboxes. Each RX Mailbox allows access to a 3 level depth FIFO, the access being offered only to the oldest received message in the FIFO.

Each mailbox consist of 4 registers.

Figure 579. CAN mailbox registers



**CAN TX mailbox identifier register (CAN\_TIxR) (x = 0..2)**

Address offsets: 0x180, 0x190, 0x1A0

Reset value: 0xXXXX XXXX (except bit 0, TXRQ = 0)

All TX registers are write protected when the mailbox is pending transmission (TMEx reset).

This register also implements the TX request control (bit 0) - reset value 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[10:0]/EXID[28:18]											EXID[17:13]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXID[12:0]													IDE	RTR	TXRQ
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 **STID[10:0]/EXID[28:18]**: Standard identifier or extended identifier

The standard identifier or the MSBs of the extended identifier (depending on the IDE bit value).

Bit 20:3 **EXID[17:0]**: Extended identifier

The LSBs of the extended identifier.

Bit 2 **IDE**: Identifier extension

This bit defines the identifier type of message in the mailbox.

0: Standard identifier.

1: Extended identifier.

Bit 1 **RTR**: Remote transmission request

0: Data frame

1: Remote frame

Bit 0 **TXRQ**: Transmit mailbox request

Set by software to request the transmission for the corresponding mailbox.

Cleared by hardware when the mailbox becomes empty.



**CAN mailbox data length control and time stamp register  
(CAN\_TDTxR) (x = 0..2)**

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x184, 0x194, 0x1A4

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLC[3:0]			
												rw	rw	rw	rw

Bits 31:16 **TIME[15:0]**: Message time stamp

This field contains the 16-bit timer value captured at the SOF transmission.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **TGT**: Transmit global time

This bit is active only when the hardware is in the Time Trigger Communication mode, TTCM bit of the CAN\_MCR register is set.

0: Time stamp TIME[15:0] is not sent.

1: Time stamp TIME[15:0] value is sent in the last two data bytes of the 8-byte message: TIME[7:0] in data byte 7 and TIME[15:8] in data byte 6, replacing the data written in CAN\_TDHxR[31:16] register (DATA6[7:0] and DATA7[7:0]). DLC must be programmed as 8 in order these two bytes to be sent over the CAN bus.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **DLC[3:0]**: Data length code

This field defines the number of data bytes a data frame contains or a remote frame request. A message can contain from 0 to 8 data bytes, depending on the value in the DLC field.

**CAN mailbox data low register (CAN\_TDLxR) (x = 0..2)**

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x188, 0x198, 0x1A8

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **DATA3[7:0]**: Data byte 3

Data byte 3 of the message.

Bits 23:16 **DATA2[7:0]**: Data byte 2

Data byte 2 of the message.

Bits 15:8 **DATA1[7:0]**: Data byte 1

Data byte 1 of the message.

Bits 7:0 **DATA0[7:0]**: Data byte 0

Data byte 0 of the message.

A message can contain from 0 to 8 data bytes and starts with byte 0.

**CAN mailbox data high register (CAN\_TDHxR) (x = 0..2)**

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x18C, 0x19C, 0x1AC

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA5[7:0]								DATA4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **DATA7[7:0]**: Data byte 7

Data byte 7 of the message.

*Note: If TGT of this message and TTCM are active, DATA7 and DATA6 will be replaced by the TIME stamp value.*

Bits 23:16 **DATA6[7:0]**: Data byte 6

Data byte 6 of the message.

Bits 15:8 **DATA5[7:0]**: Data byte 5

Data byte 5 of the message.

Bits 7:0 **DATA4[7:0]**: Data byte 4

Data byte 4 of the message.

**CAN receive FIFO mailbox identifier register (CAN\_RlRxR) (x = 0..1)**

Address offsets: 0x1B0, 0x1C0

Reset value: 0xFFFF XXXX

All RX registers are write protected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[10:0]/EXID[28:18]											EXID[17:13]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXID[12:0]													IDE	RTR	Res
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:21 **STID[10:0]/EXID[28:18]**: Standard identifier or extended identifier

The standard identifier or the MSBs of the extended identifier (depending on the IDE bit value).

Bits 20:3 **EXID[17:0]**: Extended identifier

The LSBs of the extended identifier.

Bit 2 **IDE**: Identifier extension

This bit defines the identifier type of message in the mailbox.

0: Standard identifier.

1: Extended identifier.

Bit 1 **RTR**: Remote transmission request

0: Data frame

1: Remote frame

Bit 0 Reserved, must be kept at reset value.

**CAN receive FIFO mailbox data length control and time stamp register (CAN\_RDTxR) (x = 0..1)**

Address offsets: 0x1B4, 0x1C4  
 Reset value: 0xXXXX XXXX

All RX registers are write protected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMI[7:0]								Res.	Res.	Res.	Res.	DLC[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r

Bits 31:16 **TIME[15:0]**: Message time stamp

This field contains the 16-bit timer value captured at the SOF detection.

Bits 15:8 **FMI[7:0]**: Filter match index

This register contains the index of the filter the message stored in the mailbox passed through. For more details on identifier filtering refer to [Section 49.7.4: Identifier filtering on page 1829](#) - **Filter Match Index** paragraph.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **DLC[3:0]**: Data length code

This field defines the number of data bytes a data frame contains (0 to 8). It is 0 in the case of a remote frame request.

**CAN receive FIFO mailbox data low register (CAN\_RDLxR) (x = 0..1)**

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x1B8, 0x1C8

Reset value: 0xFFFF XXXX

All RX registers are write protected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DATA3[7:0]**: Data Byte 3  
Data byte 3 of the message.

Bits 23:16 **DATA2[7:0]**: Data Byte 2  
Data byte 2 of the message.

Bits 15:8 **DATA1[7:0]**: Data Byte 1  
Data byte 1 of the message.

Bits 7:0 **DATA0[7:0]**: Data Byte 0  
Data byte 0 of the message.  
A message can contain from 0 to 8 data bytes and starts with byte 0.

**CAN receive FIFO mailbox data high register (CAN\_RDHxR) (x = 0..1)**

Address offsets: 0x1BC, 0x1CC

Reset value: 0xFFFF XXXX

All RX registers are write protected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA5[7:0]								DATA4[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DATA7[7:0]**: Data Byte 7  
Data byte 3 of the message.

Bits 23:16 **DATA6[7:0]**: Data Byte 6  
Data byte 2 of the message.

Bits 15:8 **DATA5[7:0]**: Data Byte 5  
Data byte 1 of the message.

Bits 7:0 **DATA4[7:0]**: Data Byte 4  
Data byte 0 of the message.

### 49.9.4 CAN filter registers

#### CAN filter master register (CAN\_FMR)

Address offset: 0x200  
Reset value: 0x2A1C 0E01

All bits of this register are set and cleared by software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	Res.	CANSB[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FINIT
		rw	rw	rw	rw	rw	rw								rw	

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **CANSB[5:0]**: CAN start bank

These bits are set and cleared by software. When both CAN are used, they define the start bank of each CAN interface:

000001 = 1 filter assigned to CAN1 and 27 assigned to CAN2

011011 = 27 filters assigned to CAN1 and 1 filter assigned to CAN2

- to assign all filters to one CAN set CANSB value to zero and deactivate the non used CAN
- to use CAN1 only: stop the clock on CAN2 and/or set the CAN\_MCR.INRQ on CAN2
- to use CAN2 only: set the CAN\_MCR.INRQ on CAN1 or deactivate the interrupt register CAN\_IER on CAN1

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **FINIT**: Filter initialization mode

Initialization mode for filter banks

0: Active filters mode.

1: Initialization mode for the filters.

#### CAN filter mode register (CAN\_FM1R)

Address offset: 0x204  
Reset value: 0x0000 0000

This register can be written only when the filter initialization mode is set (FINIT=1) in the CAN\_FMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	FBM27	FBM26	FBM25	FBM24	FBM23	FBM22	FBM21	FBM20	FBM19	FBM18	FBM17	FBM16
				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
FBM15	FBM14	FBM13	FBM12	FBM11	FBM10	FBM9	FBM8	FBM7	FBM6	FBM5	FBM4	FBM3	FBM2	FBM1	FBM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: Refer to [Figure 572: Filter bank scale configuration - register organization on page 1831](#).

Bits 31: Reserved, must be kept at reset value.

Bits 27:0 **FBMx**: Filter mode

Mode of the registers of Filter x.

0: Two 32-bit registers of filter bank x are in Identifier Mask mode.

1: Two 32-bit registers of filter bank x are in Identifier List mode.

### CAN filter scale register (CAN\_FS1R)

Address offset: 0x20C

Reset value: 0x0000 0000

This register can be written only when the filter initialization mode is set (FINIT=1) in the CAN\_FMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	FSC27	FSC26	FSC25	FSC24	FSC23	FSC22	FSC21	FSC20	FSC19	FSC18	FSC17	FSC16
				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
FSC15	FSC14	FSC13	FSC12	FSC11	FSC10	FSC9	FSC8	FSC7	FSC6	FSC5	FSC4	FSC3	FSC2	FSC1	FSC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:0 **FSCx**: Filter scale configuration

These bits define the scale configuration of Filters 27-0.

0: Dual 16-bit scale configuration

1: Single 32-bit scale configuration

Note: Refer to [Figure 572: Filter bank scale configuration - register organization on page 1831](#).

### CAN filter FIFO assignment register (CAN\_FFA1R)

Address offset: 0x214

Reset value: 0x0000 0000

This register can be written only when the filter initialization mode is set (FINIT=1) in the CAN\_FMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	FFA27	FFA26	FFA25	FFA24	FFA23	FFA22	FFA21	FFA20	FFA19	FFA18	FFA17	FFA16
				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
FFA15	FFA14	FFA13	FFA12	FFA11	FFA10	FFA9	FFA8	FFA7	FFA6	FFA5	FFA4	FFA3	FFA2	FFA1	FFA0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:0 **FFAx**: Filter FIFO assignment for filter x

The message passing through this filter will be stored in the specified FIFO.

0: Filter assigned to FIFO 0

1: Filter assigned to FIFO 1

### CAN filter activation register (CAN\_FA1R)

Address offset: 0x21C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	FACT 27	FACT 26	FACT 25	FACT 24	FACT 23	FACT 22	FACT 21	FACT 20	FACT 19	FACT 18	FACT 17	FACT 16
				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
FACT 15	FACT 14	FACT 13	FACT 12	FACT 11	FACT 10	FACT9	FACT8	FACT7	FACT6	FACT5	FACT4	FACT3	FACT2	FACT1	FACT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:0 **FACTx**: Filter active

The software sets this bit to activate Filter x. To modify the Filter x registers (CAN\_FxR[0:7]), the FACTx bit must be cleared or the FINIT bit of the CAN\_FMR register must be set.

0: Filter x is not active

1: Filter x is active



**Filter bank i register x (CAN\_FiRx) (i = 0..27, x = 1, 2)**

Address offsets: 0x240 to 0x31C

Reset value: 0xFFFF XXXX

There are 28 filter banks, i= 0 to 27. Each filter bank i is composed of two 32-bit registers, CAN\_FiR[2:1].

This register can only be modified when the FACTx bit of the CAN\_FAxR register is cleared or when the FINIT bit of the CAN\_FMR register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

In all configurations:

Bits 31:0 **FB[31:0]**: Filter bits

**Identifier**

Each bit of the register specifies the level of the corresponding bit of the expected identifier.

0: Dominant bit is expected

1: Recessive bit is expected

**Mask**

Each bit of the register specifies whether the bit of the associated identifier register must match with the corresponding bit of the expected identifier or not.

0: Do not care, the bit is not used for the comparison

1: Must match, the bit of the incoming identifier must have the same level has specified in the corresponding identifier register of the filter.

*Note:* Depending on the scale and mode configuration of the filter the function of each register can differ. For the filter mapping, functions description and mask registers association, refer to [Section 49.7.4: Identifier filtering on page 1829](#).

A Mask/Identifier register in **mask mode** has the same bit mapping as in **identifier list mode**.

For the register mapping/addresses of the filter banks refer to [Table 316 on page 1860](#).

### 49.9.5 bxCAN register map

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses. The registers from offset 0x200 to 0x31C are present only in CAN1.

**Table 316. bxCAN register map and reset values**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
0x000	CAN_MCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBF	RESET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.					
	Reset value																1	0								0	0	0	0	0	0	0	0					
0x004	CAN_MSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	SAMP	TXM	TXM	Res.	Res.	Res.	Res.	Res.	Res.	Res.					
	Reset value																						1	1	0	0												
0x008	CAN_TSR	LOW[2:0]		TME[2:0]			CODE[1:0]			ABRQ2	Res.	Res.	Res.	TERR2	ALST2	TXOK2	RQCP2	ABRQ1	Res.	Res.	Res.	TERR1	ALST1	TXOK1	RQCP1	ABRQ0	Res.	Res.	Res.	Res.	Res.	Res.						
	Reset value	0	0	0	1	1	1	0	0	0				0	0	0	0	0	0				0	0	0	0	0											
0x00C	CAN_RF0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.					
	Reset value																																					
0x010	CAN_RF1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.					
	Reset value																																					
0x014	CAN_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
	Reset value																																					
0x018	CAN_ESR	REC[7:0]							TEC[7:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																				
0x01C	CAN_BTR	SILM	LBKM	Res.	Res.	Res.	Res.	Res.	SJW[1:0]	Res.	TS2[2:0]		TS1[3:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRP[9:0]														
	Reset value	0	0						0	0		0	1	0	0	0	1	1																				
0x020-0x17F	-	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
0x180	CAN_TI0R	STID[10:0]/EXID[28:18]										EXID[17:0]																IDE	RTR	TXRQ								
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0				



Table 316. bxCAN register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x184	CAN_TDT0R	TIME[15:0]															Res.	Res.	Res.	Res.	Res.	Res.	Res.	TGT	Res.	Res.	Res.	Res.	DLC[3:0]				
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	-	-	-	-	-	-	-	x	-	-	-	-	x	x	x	x
0x188	CAN_TDL0R	DATA3[7:0]							DATA2[7:0]							DATA1[7:0]							DATA0[7:0]										
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x18C	CAN_TDH0R	DATA7[7:0]							DATA6[7:0]							DATA5[7:0]							DATA4[7:0]										
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x190	CAN_TI1R	STID[10:0]/EXID[28:18]										EXID[17:0]																IDE	RTR	TXRQ			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x194	CAN_TDT1R	TIME[15:0]															Res.	Res.	Res.	Res.	Res.	Res.	Res.	TGT	Res.	Res.	Res.	Res.	DLC[3:0]				
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	-	-	-	-	-	-	-	x	-	-	-	-	x	x	x	x
0x198	CAN_TDL1R	DATA3[7:0]							DATA2[7:0]							DATA1[7:0]							DATA0[7:0]										
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x19C	CAN_TDH1R	DATA7[7:0]							DATA6[7:0]							DATA5[7:0]							DATA4[7:0]										
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1A0	CAN_TI2R	STID[10:0]/EXID[28:18]										EXID[17:0]																IDE	RTR	TXRQ			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
0x1A4	CAN_TDT2R	TIME[15:0]															Res.	Res.	Res.	Res.	Res.	Res.	Res.	TGT	Res.	Res.	Res.	Res.	DLC[3:0]				
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	-	-	-	-	-	-	-	x	-	-	-	-	x	x	x	x
0x1A8	CAN_TDL2R	DATA3[7:0]							DATA2[7:0]							DATA1[7:0]							DATA0[7:0]										
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1AC	CAN_TDH2R	DATA7[7:0]							DATA6[7:0]							DATA5[7:0]							DATA4[7:0]										
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1B0	CAN_RI0R	STID[10:0]/EXID[28:18]										EXID[17:0]																IDE	RTR	Res.			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	-



Table 316. bxCAN register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1B4	<b>CAN_RDT0R</b>	TIME[15:0]															FMI[7:0]							Res	Res	Res	Res	DLC[3:0]					
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	-	-	-	-	x	x	x	x
0x1B8	<b>CAN_RDL0R</b>	DATA3[7:0]							DATA2[7:0]							DATA1[7:0]							DATA0[7:0]										
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1BC	<b>CAN_RDH0R</b>	DATA7[7:0]							DATA6[7:0]							DATA5[7:0]							DATA4[7:0]										
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1C0	<b>CAN_R11R</b>	STID[10:0]/EXID[28:18]										EXID[17:0]																IDE	RTR	Res			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	-
0x1C4	<b>CAN_RDT1R</b>	TIME[15:0]															FMI[7:0]							Res	Res	Res	Res	DLC[3:0]					
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	-	-	-	-	x	x	x	x
0x1C8	<b>CAN_RDL1R</b>	DATA3[7:0]							DATA2[7:0]							DATA1[7:0]							DATA0[7:0]										
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1CC	<b>CAN_RDH1R</b>	DATA7[7:0]							DATA6[7:0]							DATA5[7:0]							DATA4[7:0]										
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1D0-0x1FF	-	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
0x208	-	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
	-	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
0x20C	<b>CAN_FS1R</b>	FSC[27:0]																															
	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	
0x210	-	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	
0x214	<b>CAN_FFA1R</b>	FFA[27:0]																															
	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	
0x218	-	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	

Table 316. bxCAN register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x21C	CAN_FA1R	Res.	Res.	Res.	Res.	FACT[27:0]																											
	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
0x220	-	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x224-0x23F	-	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x240	CAN_F0R1	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
0x244	CAN_F0R2	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
0x248	CAN_F1R1	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
0x24C	CAN_F1R2	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
⋮	⋮	⋮																															
0x318	CAN_F27R1	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
0x31C	CAN_F27R2	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		

## 50 USB on-the-go full-speed (OTG\_FS)

### 50.1 Introduction

Portions Copyright (c) Synopsys, Inc. All rights reserved. Used with permission.

This section presents the architecture and the programming model of the OTG\_FS controller.

The following acronyms are used throughout the section:

FS	Full-speed
LS	Low-speed
MAC	Media access controller
OTG	On-the-go
PFC	Packet FIFO controller
PHY	Physical layer
USB	Universal serial bus
UTMI	USB 2.0 Transceiver Macrocell interface (UTMI)
ADP	Attach detection protocol
LPM	Link power management
BCD	Battery charging detector
HNP	Host negotiation protocol
SRP	Session request protocol

References are made to the following documents:

- USB On-The-Go Supplement, Revision 2.0
- Universal Serial Bus Revision 2.0 Specification
- USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007
- Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007
- Battery Charging Specification, Revision 1.2

The USB OTG is a dual-role device (DRD) controller that supports both device and host functions and is fully compliant with the *On-The-Go Supplement to the USB 2.0 Specification*. It can also be configured as a host-only or device-only controller, fully compliant with the *USB 2.0 Specification*. OTG\_FS supports the speeds defined in the [Table 317: OTG\\_FS speeds supported](#) below. The USB OTG supports both HNP and SRP. The only external device required is a charge pump for V<sub>BUS</sub> in OTG mode.

**Table 317. OTG\_FS speeds supported**

-	HS (480 Mb/s)	FS (12 Mb/s)	LS (1.5 Mb/s)
Host mode	-	X	X
Device mode	-	X	-

## 50.2 OTG main features

The main features can be divided into three categories: general, host-mode and device-mode features.

### 50.2.1 General features

The OTG\_FS interface general features are the following:

- It is USB-IF certified to the Universal Serial Bus Specification Rev 2.0
- It includes full support (PHY) for the optional On-The-Go (OTG) protocol detailed in the On-The-Go Supplement Rev 2.0 specification
  - Integrated support for A-B device identification (ID line)
  - Integrated support for host Negotiation protocol (HNP) and session request protocol (SRP)
  - It allows host to turn  $V_{BUS}$  off to conserve battery power in OTG applications
  - It supports OTG monitoring of  $V_{BUS}$  levels with internal comparators
  - It supports dynamic host-peripheral switch of role
- It is software-configurable to operate as:
  - SRP capable USB FS Peripheral (B-device)
  - SRP capable USB FS/LS host (A-device)
  - USB On-The-Go Full-Speed Dual Role device
- It supports FS SOF and LS Keep-alives with
  - SOF pulse PAD connectivity
  - SOF pulse internal connection to timer (TIMx)
  - Configurable framing period
  - Configurable end of frame interrupt
- It includes power saving features such as system stop during USB suspend, switch-off of clock domains internal to the digital core, PHY and DFIFO power management.
- It features a dedicated RAM of 1.25 Kbytes with advanced FIFO control:
  - Configurable partitioning of RAM space into different FIFOs for flexible and efficient use of RAM
  - Each FIFO can hold multiple packets
  - Dynamic memory allocation
  - Configurable FIFO sizes that are not powers of 2 to allow the use of contiguous memory locations
- It guarantees max USB bandwidth for up to one frame (1 ms) without system intervention.
- It supports charging port detection as described in Battery Charging Specification Revision 1.2 on the FS PHY transceiver only.
- It supports attach detection protocol (ADP).

### 50.2.2 Host-mode features

The OTG\_FS interface main features and requirements in host-mode are the following:

- External charge pump for  $V_{BUS}$  voltage generation.
- Up to 12 host channels (pipes): each channel is dynamically reconfigurable to allocate any type of USB transfer.
- Built-in hardware scheduler holding:
  - Up to 12 interrupt plus isochronous transfer requests in the periodic hardware queue
  - Up to 12 control plus bulk transfer requests in the non-periodic hardware queue
- Management of a shared Rx FIFO, a periodic Tx FIFO and a nonperiodic Tx FIFO for efficient usage of the USB data RAM.

### 50.2.3 Peripheral-mode features

The OTG\_FS interface main features in peripheral-mode are the following:

- 1 bidirectional control endpoint0
- 5 IN endpoints (EPs) configurable to support bulk, interrupt or isochronous transfers
- 5 OUT endpoints configurable to support bulk, interrupt or isochronous transfers
- Management of a shared Rx FIFO and a Tx-OUT FIFO for efficient usage of the USB data RAM
- Management of up to 6 dedicated Tx-IN FIFOs (one for each active IN EP) to put less load on the application
- Support for the soft disconnect feature.



## 50.3 OTG implementation

Table 318. OTG implementation<sup>(1)</sup>

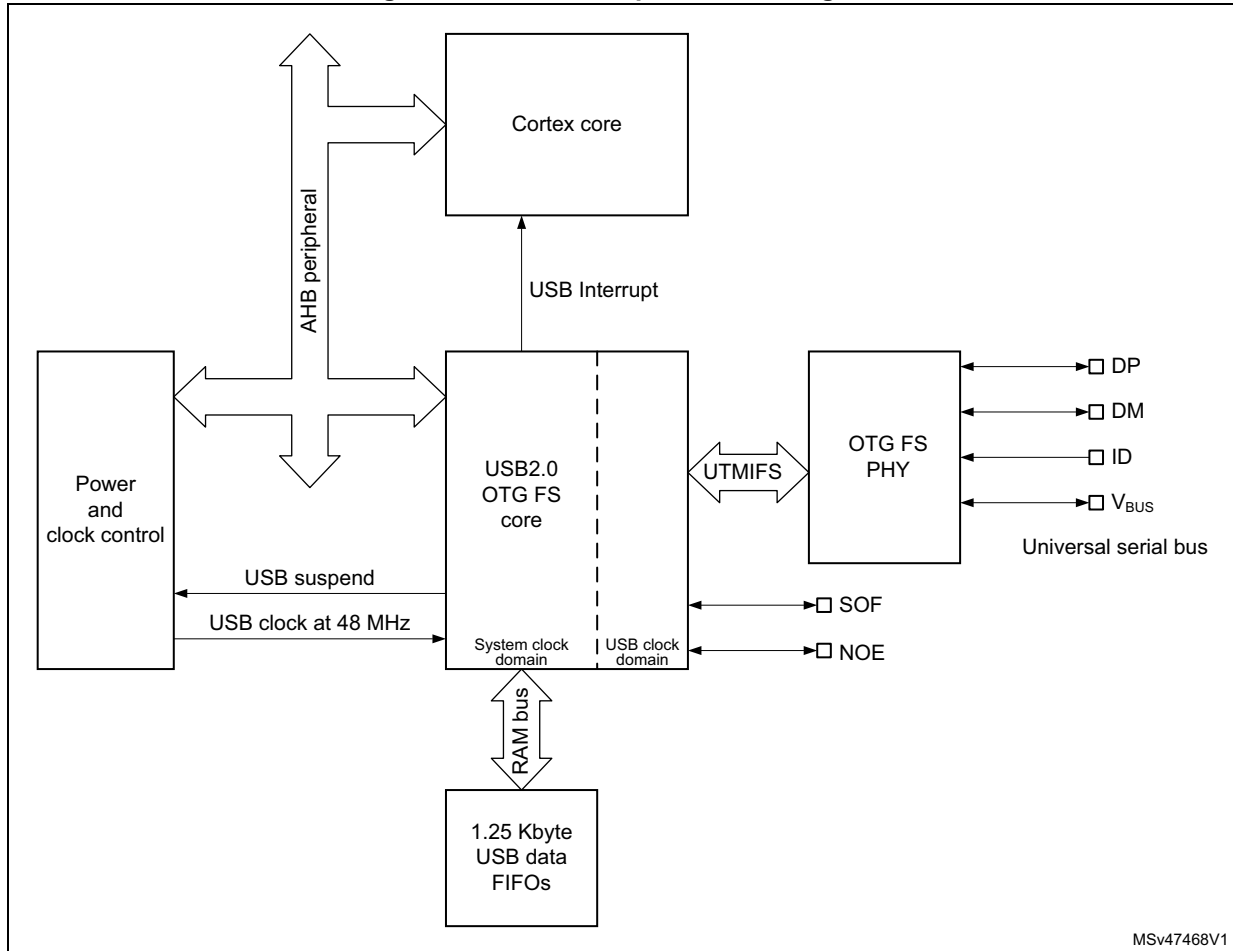
USB features	OTG_FS
Device bidirectional endpoints (including EP0)	6
Host mode channels	12
Size of dedicated SRAM	1.2 KB
USB 2.0 link power management (LPM) support	X
OTG revision supported	2.0
Attach detection protocol (ADP) support	X
Battery charging detection (BCD) support	X
PSRST / ERRATIM / STSPHST / CURMOD bits	X

1. "X" = supported, "-" not supported

## 50.4 OTG functional description

### 50.4.1 OTG block diagram

Figure 580. OTG full-speed block diagram



MSv47468V1

### 50.4.2 USB OTG pin and internal signals

Table 319. OTG\_FS input/output pins

Signal name	Signal type	Description
OTG_FS_DP	Digital input/output	USB OTG D+ line
OTG_FS_DM	Digital input/output	USB OTG D- line
OTG_FS_ID	Digital input	USB OTG ID
OTG_FS_VBUS	Analog input	USB OTG VBUS
OTG_FS_SOF	Digital output	USB OTG Start Of Frame (visibility)
OTG_FS_NOE	Digital output	USB OTG output enable for D+/D- (visibility)

Table 320. OTG\_FS input/output signals

Signal name	Signal type	Description
usb_sof	Digital output	USB OTG start-of-frame event for on chip peripherals
usb_wkup	Digital output	USB OTG wakeup event output
usb_gbl_it	Digital output	USB OTG global interrupt

### 50.4.3 OTG core

The USB OTG receives the 48 MHz clock from the reset and clock controller (RCC). The USB clock is used for driving the 48 MHz domain at full-speed (12 Mbit/s) and must be enabled prior to configuring the OTG core.

The CPU reads and writes from/to the OTG core registers through the AHB peripheral bus. It is informed of USB events through the single USB OTG interrupt line described in [Section 50.13: OTG\\_FS interrupts](#).

The CPU submits data over the USB by writing 32-bit words to dedicated OTG locations (push registers). The data are then automatically stored into Tx-data FIFOs configured within the USB data RAM. There is one Tx FIFO push register for each in-endpoint (peripheral mode) or out-channel (host mode).

The CPU receives the data from the USB by reading 32-bit words from dedicated OTG addresses (pop registers). The data are then automatically retrieved from a shared Rx FIFO configured within the 1.25-Kbyte USB data RAM. There is one Rx FIFO pop register for each out-endpoint or in-channel.

The USB protocol layer is driven by the serial interface engine (SIE) and serialized over the USB by the transceiver module within the on-chip physical layer (PHY).

### 50.4.4 Full-speed OTG PHY

The embedded full-speed OTG PHY is controlled by the OTG FS core and conveys USB control & data signals through the full-speed subset of the UTMI+ Bus (UTMIFS). It provides the physical support to USB connectivity.

The full-speed OTG PHY includes the following components:

- FS/LS transceiver module used by both host and device. It directly drives transmission and reception on the single-ended USB lines.
- Integrated ID pull-up resistor used to sample the ID line for A/B device identification.
- DP/DM integrated pull-up and pull-down resistors controlled by the OTG\_FS core depending on the current role of the device. As a peripheral, it enables the DP pull-up resistor to signal full-speed peripheral connections as soon as  $V_{BUS}$  is sensed to be at a valid level (B-session valid). In host mode, pull-down resistors are enabled on both DP/DM. Pull-up and pull-down resistors are dynamically switched when the role of the device is changed via the host negotiation protocol (HNP).
- Pull-up/pull-down resistor ECN circuit. The DP pull-up consists of two resistors controlled separately from the OTG\_FS as per the resistor Engineering Change Notice applied to USB Rev2.0. The dynamic trimming of the DP pull-up strength allows for better noise rejection and Tx/Rx signal quality.
- $V_{BUS}$  sensing comparators with hysteresis used to detect  $V_{BUS}$  valid, A-B session valid and session-end voltage thresholds. They are used to drive the session request

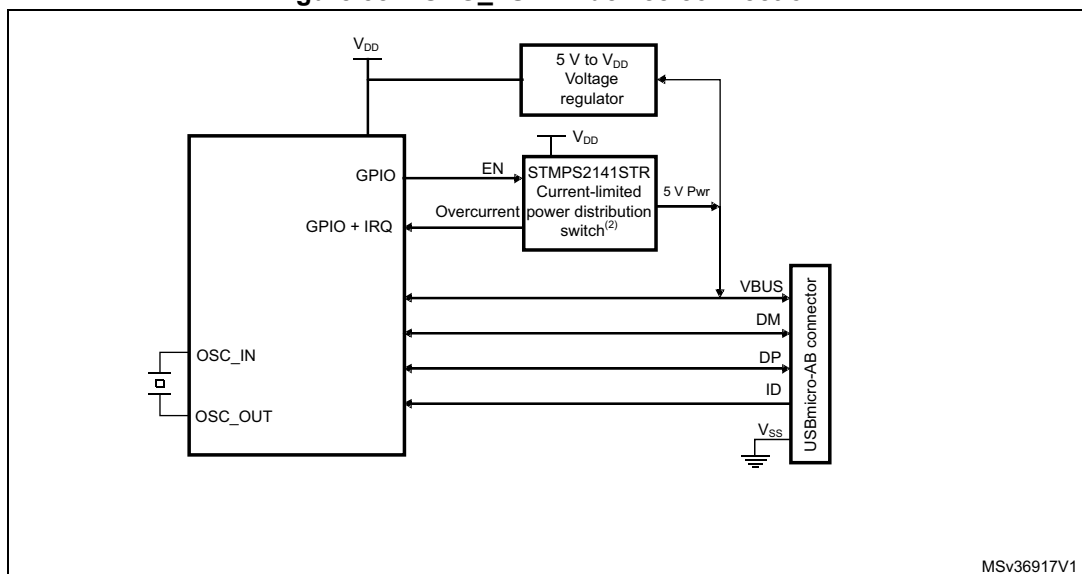
protocol (SRP), detect valid startup and end-of-session conditions, and constantly monitor the  $V_{BUS}$  supply during USB operations.

- $V_{BUS}$  pulsing method circuit used to charge/discharge  $V_{BUS}$  through resistors during the SRP (weak drive).

**Caution:** To guarantee a correct operation for the USB OTG FS peripheral, the AHB frequency should be higher than 14.2 MHz.

## 50.5 OTG dual role device (DRD)

Figure 581. OTG\_FS A-B device connection



1. External voltage regulator only needed when building a VBUS powered device.
2. STMP2141STR needed only if the application has to support a VBUS powered device. A basic power switch can be used if 5 V are available on the application board.

### 50.5.1 ID line detection

The host or peripheral (the default) role is assumed depending on the ID input pin. The ID line status is determined on plugging in the USB cable, depending on whether a MicroA or MicroB plug is connected to the micro-AB receptacle.

- If the B-side of the USB cable is connected with a floating ID wire, the integrated pull-up resistor detects a high ID level and the default peripheral role is confirmed. In this configuration the OTG\_FS complies with the standard FSM described in section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.
- If the A-side of the USB cable is connected with a grounded ID, the OTG\_FS issues an ID line status change interrupt (CIDSCHG bit in OTG\_GINTSTS) for host software initialization, and automatically switches to the host role. In this configuration the OTG\_FS complies with the standard FSM described in section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.

### 50.5.2 HNP dual role device

The HNP capable bit in the Global USB configuration register (HNPCAP bit in OTG\_GUSBCFG) enables the OTG\_FS core to dynamically change its role from A-host to A-peripheral and vice-versa, or from B-Peripheral to B-host and vice-versa according to the host negotiation protocol (HNP). The current device status can be read by the combined values of the connector ID status bit in the Global OTG control and status register (CIDSTS bit in OTG\_GOTGCTL) and the current mode of operation bit in the global interrupt and status register (CMOD bit in OTG\_GINTSTS).

The HNP program model is described in detail in [Section 50.16: OTG\\_FS programming model](#).

### 50.5.3 SRP dual role device

The SRP capable bit in the global USB configuration register (SRPCAP bit in OTG\_GUSBCFG) enables the OTG\_FS core to switch off the generation of  $V_{BUS}$  for the A-device to save power. Note that the A-device is always in charge of driving  $V_{BUS}$  regardless of the host or peripheral role of the OTG\_FS.

The SRP A/B-device program model is described in detail in [Section 50.16: OTG\\_FS programming model](#).

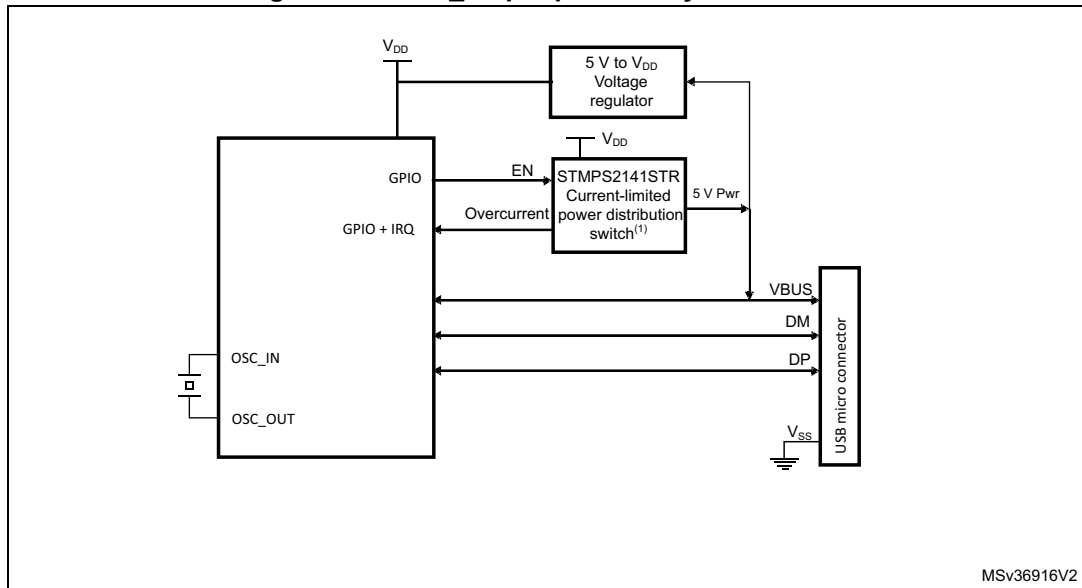
## 50.6 USB peripheral

This section gives the functional description of the OTG\_FS in the USB peripheral mode. The OTG\_FS works as an USB peripheral in the following circumstances:

- OTG B-Peripheral
  - OTG B-device default state if B-side of USB cable is plugged in
- OTG A-Peripheral
  - OTG A-device state after the HNP switches the OTG\_FS to its peripheral role
- B-device
  - If the ID line is present, functional and connected to the B-side of the USB cable, and the HNP-capable bit in the Global USB Configuration register (HNPCAP bit in OTG\_GUSBCFG) is cleared.
- Peripheral only (see [Figure 582: USB\\_FS peripheral-only connection](#))
  - The force device mode bit (FDMOD) in the [Section 50.15.4: OTG USB configuration register \(OTG\\_GUSBCFG\)](#) is set to 1, forcing the OTG\_FS core to work as an USB peripheral-only. In this case, the ID line is ignored even if it is present on the USB connector.

*Note:* To build a bus-powered device implementation in case of the B-device or peripheral-only configuration, an external regulator has to be added, that generates the necessary power-supply from  $V_{BUS}$ .

Figure 582. USB\_FS peripheral-only connection



1. Use a regulator to build a bus-powered device.

### 50.6.1 SRP-capable peripheral

The SRP capable bit in the Global USB configuration register (SRPCAP bit in OTG\_GUSBCFG) enables the OTG\_FS to support the session request protocol (SRP). In this way, it allows the remote A-device to save power by switching off  $V_{BUS}$  while the USB session is suspended.

The SRP peripheral mode program model is described in detail in the [B-device session request protocol](#) section.

### 50.6.2 Peripheral states

#### Powered state

The  $V_{BUS}$  input detects the B-session valid voltage by which the USB peripheral is allowed to enter the powered state (see USB2.0 section 9.1). The OTG\_FS then automatically connects the DP pull-up resistor to signal full-speed device connection to the host and generates the session request interrupt (SRQINT bit in OTG\_GINTSTS) to notify the powered state.

The  $V_{BUS}$  input also ensures that valid  $V_{BUS}$  levels are supplied by the host during USB operations. If a drop in  $V_{BUS}$  below B-session valid happens to be detected (for instance because of a power disturbance or if the host port has been switched off), the OTG\_FS automatically disconnects and the session end detected (SEDET bit in OTG\_GOTGINT) interrupt is generated to notify that the OTG\_FS has exited the powered state.

In the powered state, the OTG\_FS expects to receive some reset signaling from the host. No other USB operation is possible. When a reset signaling is received the reset detected interrupt (USBRST in OTG\_GINTSTS) is generated. When the reset signaling is complete, the enumeration done interrupt (ENUMDNE bit in OTG\_GINTSTS) is generated and the OTG\_FS enters the Default state.

### Soft disconnect

The powered state can be exited by software with the soft disconnect feature. The DP pull-up resistor is removed by setting the soft disconnect bit in the device control register (SDIS bit in OTG\_DCTL), causing a device disconnect detection interrupt on the host side even though the USB cable was not really removed from the host port.

### Default state

In the Default state the OTG\_FS expects to receive a SET\_ADDRESS command from the host. No other USB operation is possible. When a valid SET\_ADDRESS command is decoded on the USB, the application writes the corresponding number into the device address field in the device configuration register (DAD bit in OTG\_DCFG). The OTG\_FS then enters the address state and is ready to answer host transactions at the configured USB address.

### Suspended state

The OTG\_FS peripheral constantly monitors the USB activity. After counting 3 ms of USB idleness, the early suspend interrupt (ESUSP bit in OTG\_GINTSTS) is issued, and confirmed 3 ms later, if appropriate, by the suspend interrupt (USBSUSP bit in OTG\_GINTSTS). The device suspend bit is then automatically set in the device status register (SUSPSTS bit in OTG\_DSTS) and the OTG\_FS enters the suspended state.

The suspended state may optionally be exited by the device itself. In this case the application sets the remote wakeup signaling bit in the device control register (RWUSIG bit in OTG\_DCTL) and clears it after 1 to 15 ms.

When a resume signaling is detected from the host, the resume interrupt (WKUPINT bit in OTG\_GINTSTS) is generated and the device suspend bit is automatically cleared.

## 50.6.3 Peripheral endpoints

The OTG\_FS core instantiates the following USB endpoints:

- Control endpoint 0:
  - Bidirectional and handles control messages only
  - Separate set of registers to handle in and out transactions
  - Proper control (OTG\_DIEPCTL0/OTG\_DOEPCTL0), transfer configuration (OTG\_DIEPTSIZ0/OTG\_DOEPSIZ0), and status-interrupt (OTG\_DIEPINT0/OTG\_DOEPINT0) registers. The available set of bits inside the control and transfer size registers slightly differs from that of other endpoints
- 5 IN endpoints
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has proper control (OTG\_DIEPCTLx), transfer configuration (OTG\_DIEPTSIZx), and status-interrupt (OTG\_DIEPINTx) registers
  - The device IN endpoints common interrupt mask register (OTG\_DIEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the IN endpoints (EP0 included)
  - Support for incomplete isochronous IN transfer interrupt (IISOIXFR bit in OTG\_GINTSTS), asserted when there is at least one isochronous IN endpoint on

which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG\_GINTSTS/EOPF).

- 5 OUT endpoints
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has a proper control (OTG\_DOEPCTLx), transfer configuration (OTG\_DOEPTSIZx) and status-interrupt (OTG\_DOEPINTx) register
  - Device OUT endpoints common interrupt mask register (OTG\_DOEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the OUT endpoints (EP0 included)
  - Support for incomplete isochronous OUT transfer interrupt (INCOMPISOOUT bit in OTG\_GINTSTS), asserted when there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG\_GINTSTS/EOPF).

### Endpoint control

- The following endpoint controls are available to the application through the device endpoint-x IN/OUT control register (OTG\_DIEPCTLx/OTG\_DOEPCTLx):
  - Endpoint enable/disable
  - Endpoint activate in current configuration
  - Program USB transfer type (isochronous, bulk, interrupt)
  - Program supported packet size
  - Program Tx FIFO number associated with the IN endpoint
  - Program the expected or transmitted data0/data1 PID (bulk/interrupt only)
  - Program the even/odd frame during which the transaction is received or transmitted (isochronous only)
  - Optionally program the NAK bit to always negative-acknowledge the host regardless of the FIFO status
  - Optionally program the STALL bit to always stall host tokens to that endpoint
  - Optionally program the SNOOP mode for OUT endpoint not to check the CRC field of received data

### Endpoint transfer

The device endpoint-x transfer size registers (OTG\_DIEPTSIZx/OTG\_DOEPTSIZx) allow the application to program the transfer size parameters and read the transfer status. Programming must be done before setting the endpoint enable bit in the endpoint control register. Once the endpoint is enabled, these fields are read-only as the OTG\_FS core updates them with the current transfer status.

The following transfer parameters can be programmed:

- Transfer size in bytes
- Number of packets that constitute the overall transfer size

### Endpoint status/interrupt

The device endpoint-x interrupt registers (OTG\_DIEPINTx/OTG\_DOEPINTx) indicate the status of an endpoint with respect to USB- and AHB-related events. The application must read these registers when the OUT endpoint interrupt bit or the IN endpoint interrupt bit in



the core interrupt register (OEPINT bit in OTG\_GINTSTS or IEPINT bit in OTG\_GINTSTS, respectively) is set. Before the application can read these registers, it must first read the device all endpoints interrupt (OTG\_DAINTE) register to get the exact endpoint number for the device endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG\_DAINTE and OTG\_GINTSTS registers

The peripheral core provides the following status checks and interrupt generation:

- Transfer completed interrupt, indicating that data transfer was completed on both the application (AHB) and USB sides
- Setup stage has been done (control-out only)
- Associated transmit FIFO is half or completely empty (in endpoints)
- NAK acknowledge has been transmitted to the host (isochronous-in only)
- IN token received when Tx FIFO was empty (bulk-in/interrupt-in only)
- Out token received when endpoint was not yet enabled
- Babble error condition has been detected
- Endpoint disable by application is effective
- Endpoint NAK by application is effective (isochronous-in only)
- More than 3 back-to-back setup packets were received (control-out only)
- Timeout condition detected (control-in only)
- Isochronous out packet has been dropped, without generating an interrupt

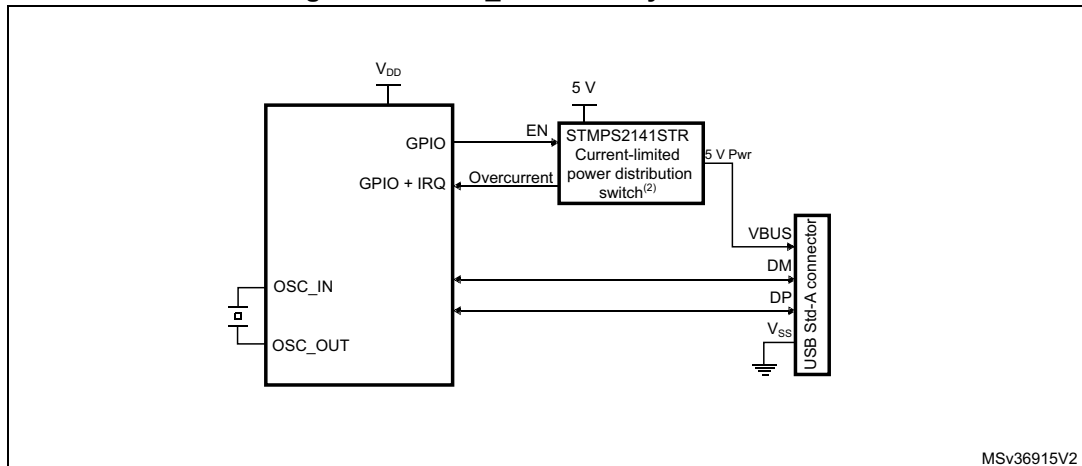
## 50.7 USB host

This section gives the functional description of the OTG\_FS in the USB host mode. The OTG\_FS works as a USB host in the following circumstances:

- OTG A-host
  - OTG A-device default state when the A-side of the USB cable is plugged in
- OTG B-host
  - OTG B-device after HNP switching to the host role
- A-device
  - If the ID line is present, functional and connected to the A-side of the USB cable, and the HNP-capable bit is cleared in the Global USB Configuration register (HNPCAP bit in OTG\_GUSBCFG). Integrated pull-down resistors are automatically set on the DP/DM lines.
- Host only
  - The force host mode bit (FHMOD) in the [OTG USB configuration register \(OTG\\_GUSBCFG\)](#) forces the OTG\_FS core to work as a USB host-only. In this case, the ID line is ignored even if present on the USB connector. Integrated pull-down resistors are automatically set on the DP/DM lines.

*Note:* On-chip 5 V  $V_{BUS}$  generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch must be added externally to drive the 5 V  $V_{BUS}$  line. The external charge pump can be driven by any GPIO output. This is required for the OTG A-host, A-device and host-only configurations.

Figure 583. USB\_FS host-only connection



1.  $V_{DD}$  range is between 2 V and 3.6 V.

### 50.7.1 SRP-capable host

SRP support is available through the SRP capable bit in the global USB configuration register (SRPCAP bit in OTG\_GUSBCFG). With the SRP feature enabled, the host can save power by switching off the  $V_{BUS}$  power while the USB session is suspended.

The SRP host mode program model is described in detail in the [A-device session request protocol](#) section.

### 50.7.2 USB host states

#### Host port power

On-chip 5 V  $V_{BUS}$  generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch, must be added externally to drive the 5 V  $V_{BUS}$  line. The external charge pump can be driven by any GPIO output or via an I<sup>2</sup>C interface connected to an external PMIC (power management IC). When the application decides to power on  $V_{BUS}$ , it must also set the port power bit in the host port control and status register (PPWR bit in OTG\_HPRT).

#### $V_{BUS}$ valid

When HNP or SRP is enabled the VBUS sensing pin should be connected to  $V_{BUS}$ . The  $V_{BUS}$  input ensures that valid  $V_{BUS}$  levels are supplied by the charge pump during USB operations. Any unforeseen  $V_{BUS}$  voltage drop below the  $V_{BUS}$  valid threshold (4.4 V) leads to an OTG interrupt triggered by the session end detected bit (SEDET bit in OTG\_GOTGINT). The application is then required to remove the  $V_{BUS}$  power and clear the port power bit.

When HNP and SRP are both disabled, the VBUS sensing pin does not need to be connected to  $V_{BUS}$ .

The charge pump overcurrent flag can also be used to prevent electrical damage. Connect the overcurrent flag output from the charge pump to any GPIO input and configure it to generate a port interrupt on the active level. The overcurrent ISR must promptly disable the  $V_{BUS}$  generation and clear the port power bit.

### Host detection of a peripheral connection

If SRP or HNP are enabled, even if USB peripherals or B-devices can be attached at any time, the OTG\_FS will not detect any bus connection until  $V_{BUS}$  is no longer sensed at a valid level (5 V). When  $V_{BUS}$  is at a valid level and a remote B-device is attached, the OTG\_FS core issues a host port interrupt triggered by the device connected bit in the host port control and status register (PCDET bit in OTG\_HPRT).

When HNP and SRP are both disabled, USB peripherals or B-device are detected as soon as they are connected. The OTG\_FS core issues a host port interrupt triggered by the device connected bit in the host port control and status (PCDET bit in OTG\_HPRT).

### Host detection of peripheral a disconnection

The peripheral disconnection event triggers the disconnect detected interrupt (DISCINT bit in OTG\_GINTSTS).

### Host enumeration

After detecting a peripheral connection the host must start the enumeration process by sending USB reset and configuration commands to the new peripheral.

Before starting to drive a USB reset, the application waits for the OTG interrupt triggered by the debounce done bit (DBCDNE bit in OTG\_GOTGINT), which indicates that the bus is stable again after the electrical debounce caused by the attachment of a pull-up resistor on DP (FS) or DM (LS).

The application drives a USB reset signaling (single-ended zero) over the USB by keeping the port reset bit set in the host port control and status register (PRST bit in OTG\_HPRT) for a minimum of 10 ms and a maximum of 20 ms. The application takes care of the timing count and then of clearing the port reset bit.

Once the USB reset sequence has completed, the host port interrupt is triggered by the port enable/disable change bit (PENCHNG bit in OTG\_HPRT). This informs the application that the speed of the enumerated peripheral can be read from the port speed field in the host port control and status register (PSPD bit in OTG\_HPRT) and that the host is starting to drive SOFs (FS) or Keep alives (LS). The host is now ready to complete the peripheral enumeration by sending peripheral configuration commands.

### Host suspend

The application decides to suspend the USB activity by setting the port suspend bit in the host port control and status register (PSUSP bit in OTG\_HPRT). The OTG\_FS core stops sending SOFs and enters the suspended state.

The suspended state can be optionally exited on the remote device's initiative (remote wakeup). In this case the remote wakeup interrupt (WKUPINT bit in OTG\_GINTSTS) is generated upon detection of a remote wakeup signaling, the port resume bit in the host port control and status register (PRES bit in OTG\_HPRT) self-sets, and resume signaling is automatically driven over the USB. The application must time the resume window and then clear the port resume bit to exit the suspended state and restart the SOF.

If the suspended state is exited on the host initiative, the application must set the port resume bit to start resume signaling on the host port, time the resume window and finally clear the port resume bit.

### 50.7.3 Host channels

The OTG\_FS core instantiates 12 host channels. Each host channel supports an USB host transfer (USB pipe). The host is not able to support more than 12 transfer requests at the same time. If more than 12 transfer requests are pending from the application, the host controller driver (HCD) must re-allocate channels when they become available from previous duty, that is, after receiving the transfer completed and channel halted interrupts.

Each host channel can be configured to support in/out and any type of periodic/nonperiodic transaction. Each host channel makes use of proper control (OTG\_HCCHARx), transfer configuration (OTG\_HCTSIZx) and status/interrupt (OTG\_HCINTx) registers with associated mask (OTG\_HCINTMSKx) registers.

#### Host channel control

- The following host channel controls are available to the application through the host channel-x characteristics register (OTG\_HCCHARx):
  - Channel enable/disable
  - Program the FS/LS speed of target USB peripheral
  - Program the address of target USB peripheral
  - Program the endpoint number of target USB peripheral
  - Program the transfer IN/OUT direction
  - Program the USB transfer type (control, bulk, interrupt, isochronous)
  - Program the maximum packet size (MPS)
  - Program the periodic transfer to be executed during odd/even frames

#### Host channel transfer

The host channel transfer size registers (OTG\_HCTSIZx) allow the application to program the transfer size parameters, and read the transfer status. Programming must be done before setting the channel enable bit in the host channel characteristics register. Once the endpoint is enabled the packet count field is read-only as the OTG\_FS core updates it according to the current transfer status.

- The following transfer parameters can be programmed:
  - transfer size in bytes
  - number of packets making up the overall transfer size
  - initial data PID

#### Host channel status/interrupt

The host channel-x interrupt register (OTG\_HCINTx) indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read these register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG\_GINTSTS) is set. Before the application can read these registers, it must first read the host all channels interrupt (OTG\_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG\_HAINT and OTG\_GINTSTS registers.

The mask bits for each interrupt source of each channel are also available in the OTG\_HCINTMSKx register.

- The host core provides the following status checks and interrupt generation:
  - Transfer completed interrupt, indicating that the data transfer is complete on both the application (AHB) and USB sides
  - Channel has stopped due to transfer completed, USB transaction error or disable command from the application
  - Associated transmit FIFO is half or completely empty (IN endpoints)
  - ACK response received
  - NAK response received
  - STALL response received
  - USB transaction error due to CRC failure, timeout, bit stuff error, false EOP
  - Babble error
  - frame overrun
  - data toggle error

#### 50.7.4 Host scheduler

The host core features a built-in hardware scheduler which is able to autonomously re-order and manage the USB transaction requests posted by the application. At the beginning of each frame the host executes the periodic (isochronous and interrupt) transactions first, followed by the nonperiodic (control and bulk) transactions to achieve the higher level of priority granted to the isochronous and interrupt transfer types by the USB specification.

The host processes the USB transactions through request queues (one for periodic and one for nonperiodic). Each request queue can hold up to 8 entries. Each entry represents a pending transaction request from the application, and holds the IN or OUT channel number along with other information to perform a transaction on the USB. The order in which the requests are written to the queue determines the sequence of the transactions on the USB interface.

At the beginning of each frame, the host processes the periodic request queue first, followed by the nonperiodic request queue. The host issues an incomplete periodic transfer interrupt (IPXFR bit in OTG\_GINTSTS) if an isochronous or interrupt transaction scheduled for the current frame is still pending at the end of the current frame. The OTG\_FS core is fully responsible for the management of the periodic and nonperiodic request queues. The periodic transmit FIFO and queue status register (OTG\_HPTXSTS) and nonperiodic transmit FIFO and queue status register (OTG\_HNPTXSTS) are read-only registers which can be used by the application to read the status of each request queue. They contain:

- The number of free entries currently available in the periodic (nonperiodic) request queue (8 max)
- Free space currently available in the periodic (nonperiodic) Tx FIFO (out-transactions)
- IN/OUT token, host channel number and other status information.

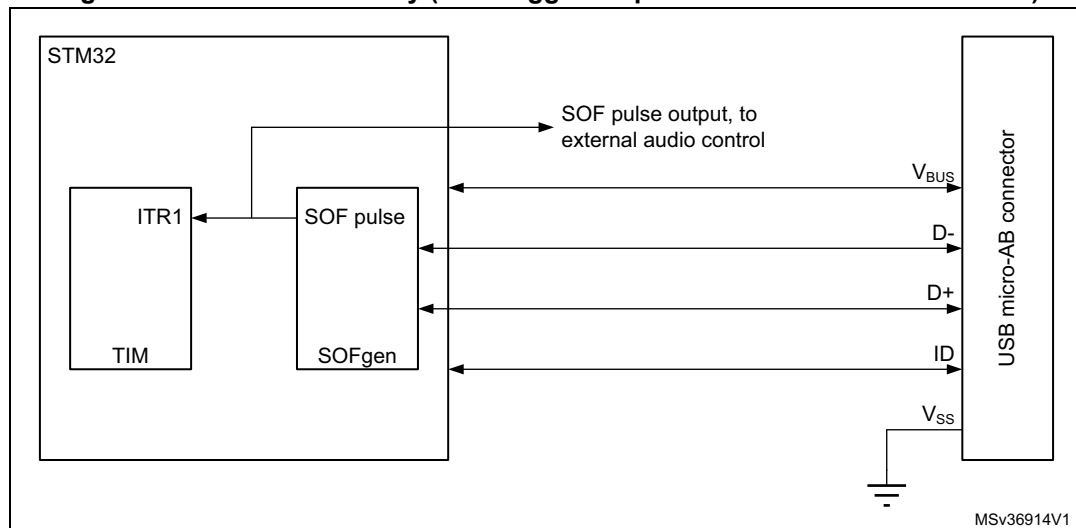
As request queues can hold a maximum of 8 entries each, the application can push to schedule host transactions in advance with respect to the moment they physically reach the SB for a maximum of 8 pending periodic transactions plus 8 pending non-periodic transactions.

To post a transaction request to the host scheduler (queue) the application must check that there is at least 1 entry available in the periodic (nonperiodic) request queue by reading the

PTXQSAV bits in the OTG\_HNPTXSTS register or NPTQXSAV bits in the OTG\_HNPTXSTS register.

## 50.8 SOF trigger

Figure 584. SOF connectivity (SOF trigger output to TIM and ITR1 connection)



The OTG\_FS core provides means to monitor, track and configure SOF framing in the host and peripheral, as well as an SOF pulse output connectivity feature.

Such utilities are especially useful for adaptive audio clock generation techniques, where the audio peripheral needs to synchronize to the isochronous stream provided by the PC, or the host needs to trim its framing rate according to the requirements of the audio peripheral.

### 50.8.1 Host SOFs

In host mode the number of PHY clocks occurring between the generation of two consecutive SOF (FS) or Keep-alive (LS) tokens is programmable in the host frame interval register (HFIR), thus providing application control over the SOF framing period. An interrupt is generated at any start of frame (SOF bit in OTG\_GINTSTS). The current frame number and the time remaining until the next SOF are tracked in the host frame number register (HFNUM).

A SOF pulse signal, is generated at any SOF starting token and with a width of 20 HCLK cycles. The SOF pulse is also internally connected to the input trigger of the timer, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.

### 50.8.2 Peripheral SOFs

In device mode, the start of frame interrupt is generated each time an SOF token is received on the USB (SOF bit in OTG\_GINTSTS). The corresponding frame number can be read from the device status register (FNSOF bit in OTG\_DSTS). A SOF pulse signal with a width of 20 HCLK cycles is also generated. The SOF pulse signal is also internally connected to the TIM input trigger, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.

The end of periodic frame interrupt (OTG\_GINTSTS/EOPF) is used to notify the application when 80%, 85%, 90% or 95% of the time frame interval elapsed depending on the periodic frame interval field in the device configuration register (PFIVL bit in OTG\_DCFG). This feature can be used to determine if all of the isochronous traffic for that frame is complete.

## 50.9 OTG low-power modes

Table 321 below defines the STM32 low power modes and their compatibility with the OTG.

**Table 321. Compatibility of STM32 low power modes with the OTG**

Mode	Description	USB compatibility
Run	MCU fully active	Required when USB not in suspend state.
Sleep	USB suspend exit causes the device to exit Sleep mode. Peripheral registers content is kept.	Available while USB is in suspend state.
Stop	USB suspend exit causes the device to exit Stop mode. Peripheral registers content is kept <sup>(1)</sup> .	Available while USB is in suspend state.
Standby	Powered-down. The peripheral must be reinitialized after exiting Standby mode.	Not compatible with USB applications.

1. Within Stop mode there are different possible settings. Some restrictions may also exist, please refer to [Section 5: Power control \(PWR\)](#) to understand which (if any) restrictions apply when using OTG.

The following bits and procedures reduce power consumption.

The power consumption of the OTG PHY is controlled by two or three bits in the general core configuration register, depending on OTG revision supported.

- PHY power down (OTG\_GCCFG/PWRDWN)  
It switches on/off the full-speed transceiver module of the PHY. It must be preliminarily set to allow any USB operation
- V<sub>BUS</sub> detection enable (OTG\_GCCFG/VBDEN)  
It switches on/off the V<sub>BUS</sub> sensing comparators associated with OTG operations

Power reduction techniques are available while in the USB suspended state, when the USB session is not yet valid or the device is disconnected.

- Stop PHY clock (STPPCLK bit in OTG\_PCGCCTL)  
When setting the stop PHY clock bit in the clock gating control register, most of the 48 MHz clock domain internal to the OTG full-speed core is switched off by clock gating. The dynamic power consumption due to the USB clock switching activity is cut even if the 48 MHz clock input is kept running by the application  
Most of the transceiver is also disabled, and only the part in charge of detecting the asynchronous resume or remote wakeup event is kept alive.
- Gate HCLK (GATEHCLK bit in OTG\_PCGCCTL)  
When setting the Gate HCLK bit in the clock gating control register, most of the system clock domain internal to the OTG\_FS core is switched off by clock gating. Only the register read and write interface is kept alive. The dynamic power consumption due to



the USB clock switching activity is cut even if the system clock is kept running by the application for other purposes.

- USB system stop

When the OTG\_FS is in the USB suspended state, the application may decide to drastically reduce the overall power consumption by a complete shut down of all the clock sources in the system. USB System Stop is activated by first setting the Stop PHY clock bit and then configuring the system deep sleep mode in the power control system module (PWR).

The OTG\_FS core automatically reactivates both system and USB clocks by asynchronous detection of remote wakeup (as an host) or resume (as a device) signaling on the USB.

To save dynamic power, the USB data FIFO is clocked only when accessed by the OTG\_FS core.

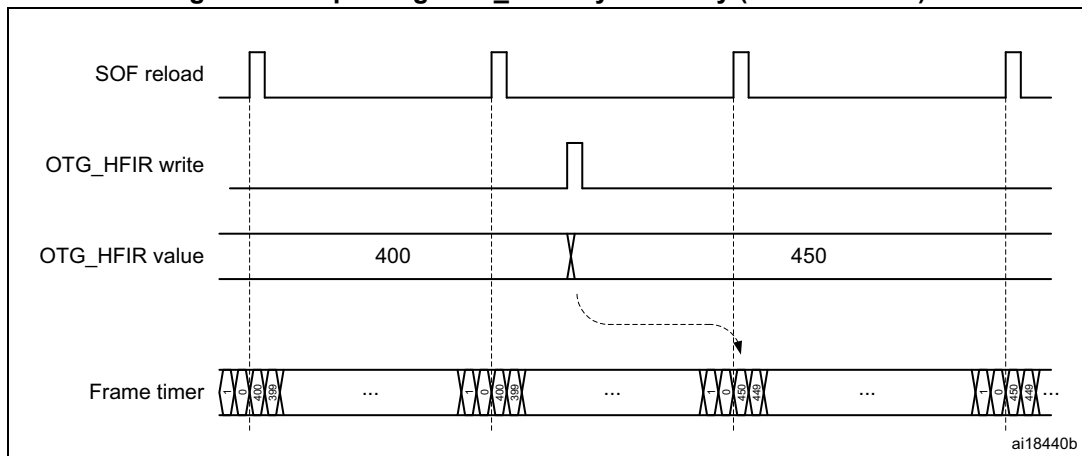
### 50.10 Dynamic update of the OTG\_HFIR register

The USB core embeds a dynamic trimming capability of SOF framing period in host mode allowing to synchronize an external device with the SOF frames.

When the OTG\_HFIR register is changed within a current SOF frame, the SOF period correction is applied in the next frame as described in [Figure 585](#).

For a dynamic update, it is required to set RLDCTRL=0.

**Figure 585. Updating OTG\_HFIR dynamically (RLDCTRL = 0)**



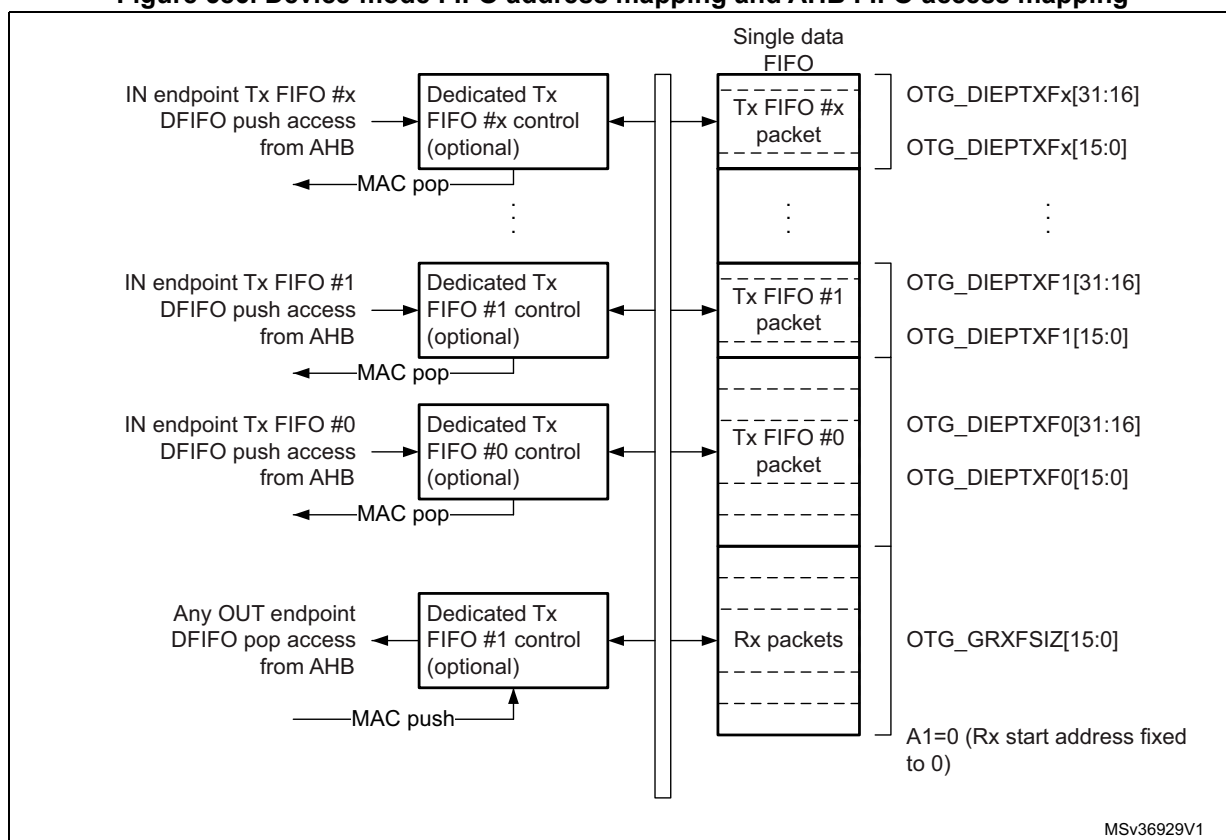
### 50.11 USB data FIFOs

The USB system features 1.25 Kbytes of dedicated RAM with a sophisticated FIFO control mechanism. The packet FIFO controller module in the OTG\_FS core organizes RAM space into Tx FIFOs into which the application pushes the data to be temporarily stored before the USB transmission, and into a single Rx FIFO where the data received from the USB are temporarily stored before retrieval (popped) by the application. The number of instructed FIFOs and how these are organized inside the RAM depends on the device's role. In peripheral mode an additional Tx FIFO is instructed for each active IN endpoint. Any FIFO size is software configured to better meet the application requirements.



### 50.11.1 Peripheral FIFO architecture

Figure 586. Device-mode FIFO address mapping and AHB FIFO access mapping



#### Peripheral Rx FIFO

The OTG peripheral uses a single receive FIFO that receives the data directed to all OUT endpoints. Received packets are stacked back-to-back until free space is available in the Rx FIFO. The status of the received packet (which contains the OUT endpoint destination number, the byte count, the data PID and the validity of the received data) is also stored by the core on top of the data payload. When no more space is available, host transactions are NACKed and an interrupt is received on the addressed endpoint. The size of the receive FIFO is configured in the receive FIFO size register (OTG\_GRXFSIZ).

The single receive FIFO architecture makes it more efficient for the USB peripheral to fill in the receive RAM buffer:

- All OUT endpoints share the same RAM buffer (shared FIFO)
- The OTG\_FS core can fill in the receive FIFO up to the limit for any host sequence of OUT tokens

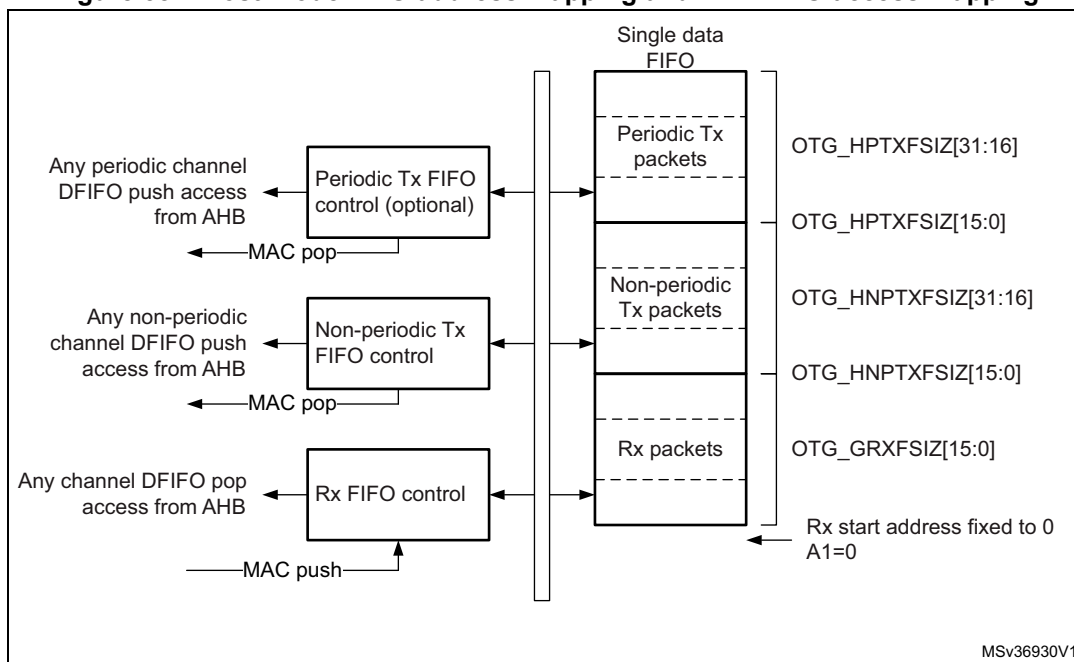
The application keeps receiving the Rx FIFO non-empty interrupt (RXFLVL bit in OTG\_GINTSTS) as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register (OTG\_GRXSTSP) and finally pops data off the receive FIFO by reading from the endpoint-related pop address.

### Peripheral Tx FIFOs

The core has a dedicated FIFO for each IN endpoint. The application configures FIFO sizes by writing the endpoint 0 transmit FIFO size register (OTG\_DIEPTXF0) for IN endpoint0 and the device IN endpoint transmit FIFOx registers (OTG\_DIEPTXFx) for IN endpoint-x.

### 50.11.2 Host FIFO architecture

Figure 587. Host-mode FIFO address mapping and AHB FIFO access mapping



### Host Rx FIFO

The host uses one receiver FIFO for all periodic and nonperiodic transactions. The FIFO is used as a receive buffer to hold the received data (payload of the received packet) from the USB until it is transferred to the system memory. Packets received from any remote IN endpoint are stacked back-to-back until free space is available. The status of each received packet with the host channel destination, byte count, data PID and validity of the received data are also stored into the FIFO. The size of the receive FIFO is configured in the receive FIFO size register (OTG\_GRXF0SIZ).

The single receive FIFO architecture makes it highly efficient for the USB host to fill in the receive data buffer:

- All IN configured host channels share the same RAM buffer (shared FIFO)
- The OTG\_FS core can fill in the receive FIFO up to the limit for any sequence of IN tokens driven by the host software

The application receives the Rx FIFO not-empty interrupt as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register and finally pops the data off the receive FIFO.

## Host Tx FIFOs

The host uses one transmit FIFO for all non-periodic (control and bulk) OUT transactions and one transmit FIFO for all periodic (isochronous and interrupt) OUT transactions. FIFOs are used as transmit buffers to hold the data (payload of the transmit packet) to be transmitted over the USB. The size of the periodic (nonperiodic) Tx FIFO is configured in the host periodic (nonperiodic) transmit FIFO size OTG\_HPTXFSIZ / OTG\_HNPTXFSIZ register.

The two Tx FIFO implementation derives from the higher priority granted to the periodic type of traffic over the USB frame. At the beginning of each frame, the built-in host scheduler processes the periodic request queue first, followed by the nonperiodic request queue.

The two transmit FIFO architecture provides the USB host with separate optimization for periodic and nonperiodic transmit data buffer management:

- All host channels configured to support periodic (nonperiodic) transactions in the OUT direction share the same RAM buffer (shared FIFOs)
- The OTG\_FS core can fill in the periodic (nonperiodic) transmit FIFO up to the limit for any sequence of OUT tokens driven by the host software

The OTG\_FS core issues the periodic Tx FIFO empty interrupt (PTXFE bit in OTG\_GINTSTS) as long as the periodic Tx FIFO is half or completely empty, depending on the value of the periodic Tx FIFO empty level bit in the AHB configuration register (PTXFELVL bit in OTG\_GAHBCFG). The application can push the transmission data in advance as long as free space is available in both the periodic Tx FIFO and the periodic request queue. The host periodic transmit FIFO and queue status register (OTG\_HPTXSTS) can be read to know how much space is available in both.

OTG\_FS core issues the non periodic Tx FIFO empty interrupt (NPTXFE bit in OTG\_GINTSTS) as long as the nonperiodic Tx FIFO is half or completely empty depending on the non periodic Tx FIFO empty level bit in the AHB configuration register (TXFELVL bit in OTG\_GAHBCFG). The application can push the transmission data as long as free space is available in both the nonperiodic Tx FIFO and nonperiodic request queue. The host nonperiodic transmit FIFO and queue status register (OTG\_HNPTXSTS) can be read to know how much space is available in both.

### 50.11.3 FIFO RAM allocation

#### Device mode

**Receive FIFO RAM allocation:** the application should allocate RAM for SETUP packets:

- 10 locations must be reserved in the receive FIFO to receive SETUP packets on control endpoint. The core does not use these locations, which are reserved for SETUP packets, to write any other data.
- One location is to be allocated for Global OUT NAK.
- Status information is written to the FIFO along with each received packet. Therefore, a minimum space of  $(\text{largest packet size} / 4) + 1$  must be allocated to receive packets. If multiple isochronous endpoints are enabled, then at least two  $(\text{largest packet size} / 4) + 1$  spaces must be allocated to receive back-to-back packets. Typically, two  $(\text{largest packet size} / 4) + 1$  spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.
- Along with the last packet for each endpoint, transfer complete status information is also pushed to the FIFO. One location for each OUT endpoint is recommended.

Device RxFIFO =

$(5 * \text{number of control endpoints} + 8) + ((\text{largest USB packet used} / 4) + 1 \text{ for status information}) + (2 * \text{number of OUT endpoints}) + 1 \text{ for Global NAK}$

Example: The MPS is 1,024 bytes for a periodic USB packet and 512 bytes for a non-periodic USB packet. There are three OUT endpoints, three IN endpoints, one control endpoint, and three host channels.

Device RxFIFO =  $(5 * 1 + 8) + ((1,024 / 4) + 1) + (2 * 4) + 1 = 279$

**Transmit FIFO RAM allocation:** the minimum RAM space required for each IN endpoint Transmit FIFO is the maximum packet size for that particular IN endpoint.

*Note: More space allocated in the transmit IN endpoint FIFO results in better performance on the USB.*

### Host mode

Receive FIFO RAM allocation:

Status information is written to the FIFO along with each received packet. Therefore, a minimum space of  $(\text{largest packet size} / 4) + 1$  must be allocated to receive packets. If multiple isochronous channels are enabled, then at least two  $(\text{largest packet size} / 4) + 1$  spaces must be allocated to receive back-to-back packets. Typically, two  $(\text{largest packet size} / 4) + 1$  spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.

Along with the last packet in the host channel, transfer complete status information is also pushed to the FIFO. So one location must be allocated for this.

Host RxFIFO =  $(\text{largest USB packet used} / 4) + 1 \text{ for status information} + 1 \text{ transfer complete}$

Example: Host RxFIFO =  $((1,024 / 4) + 1) + 1 = 258$

Transmit FIFO RAM allocation:

The minimum amount of RAM required for the host Non-periodic Transmit FIFO is the largest maximum packet size among all supported non-periodic OUT channels.

Typically, two largest packet sizes worth of space is recommended, so that when the current packet is under transfer to the USB, the CPU can get the next packet.

Non-Periodic TxFIFO =  $\text{largest non-periodic USB packet used} / 4$

Example: Non-Periodic TxFIFO =  $(512 / 4) = 128$

The minimum amount of RAM required for host periodic Transmit FIFO is the largest maximum packet size out of all the supported periodic OUT channels. If there is at least one isochronous OUT endpoint, then the space must be at least two times the maximum packet size of that channel.

Host Periodic TxFIFO =  $\text{largest periodic USB packet used} / 4$

Example: Host Periodic TxFIFO =  $(1,024 / 4) = 256$

*Note: More space allocated in the Transmit Non-periodic FIFO results in better performance on the USB.*

## 50.12 OTG\_FS system performance

Best USB and system performance is achieved owing to the large RAM buffers, the highly configurable FIFO sizes, the quick 32-bit FIFO access through AHB push/pop registers and, especially, the advanced FIFO control mechanism. Indeed, this mechanism allows the OTG\_FS to fill in the available RAM space at best regardless of the current USB sequence. With these features:

- The application gains good margins to calibrate its intervention in order to optimize the CPU bandwidth usage:
  - It can accumulate large amounts of transmission data in advance compared to when they are effectively sent over the USB
  - It benefits of a large time margin to download data from the single receive FIFO
- The USB core is able to maintain its full operating rate, that is to provide maximum full-speed bandwidth with a great margin of autonomy versus application intervention:
  - It has a large reserve of transmission data at its disposal to autonomously manage the sending of data over the USB
  - It has a lot of empty space available in the receive buffer to autonomously fill it in with the data coming from the USB

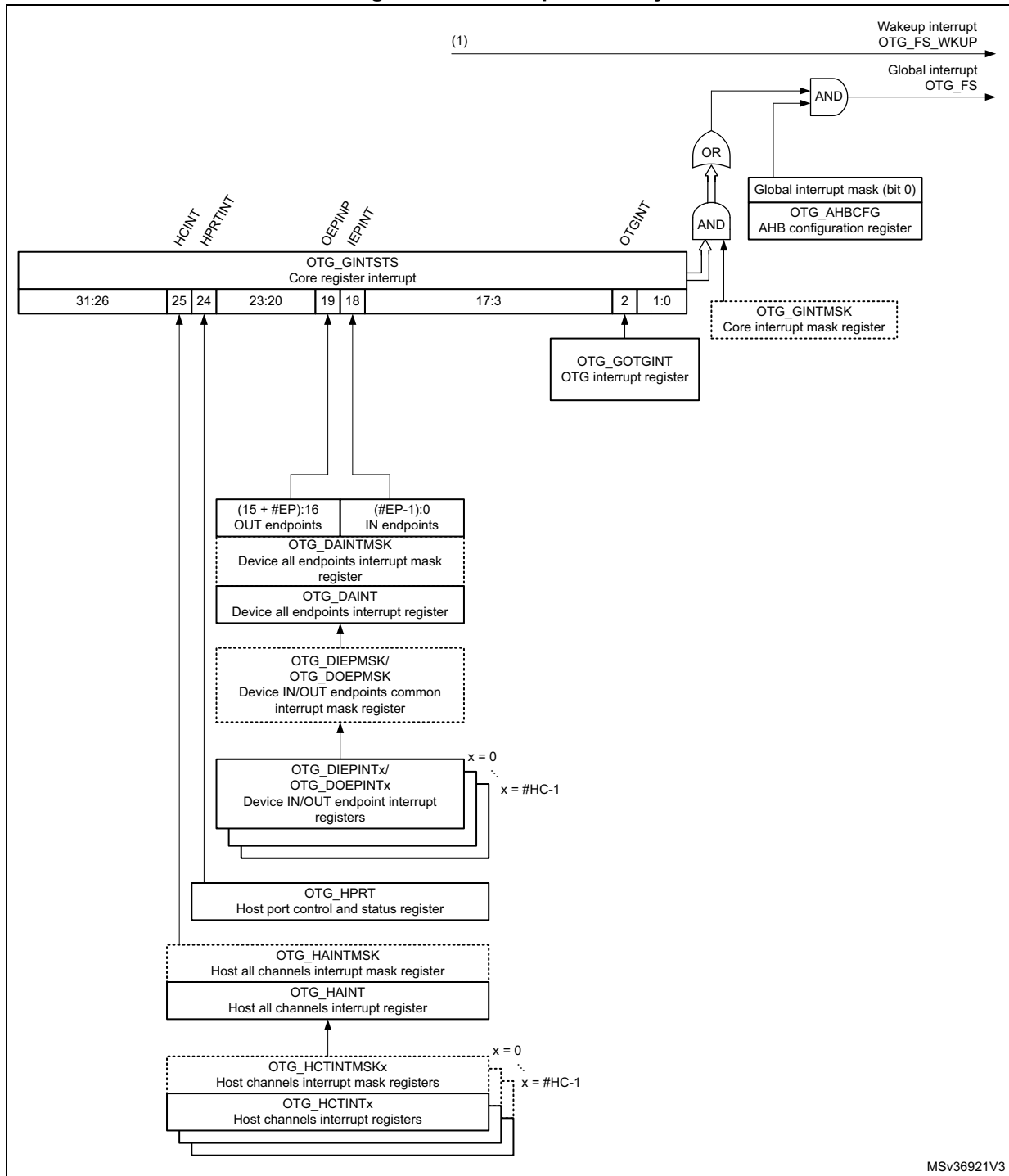
As the OTG\_FS core is able to fill in the 1.25-Kbyte RAM buffer very efficiently, and as 1.25-Kbyte of transmit/receive data is more than enough to cover a full speed frame, the USB system is able to withstand the maximum full-speed data rate for up to one USB frame (1 ms) without any CPU intervention.

## 50.13 OTG\_FS interrupts

When the OTG\_FS controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG\_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

[Figure 588](#) shows the interrupt hierarchy.

Figure 588. Interrupt hierarchy



MSv36921V3

1. OTG\_FS\_WKUP becomes active (high state) when resume condition occurs during L1 SLEEP or L2 SUSPEND states.

## 50.14 OTG\_FS control and status registers

By reading from and writing to the control and status registers (CSRs) through the AHB slave interface, the application controls the OTG\_FS controller. These registers are 32 bits wide, and the addresses are 32-bit block aligned. The OTG\_FS registers must be accessed by words (32 bits).

CSRs are classified as follows:

- Core global registers
- Host-mode registers
- Host global registers
- Host port CSRs
- Host channel-specific registers
- Device-mode registers
- Device global registers
- Device endpoint-specific registers
- Power and clock-gating registers
- Data FIFO (DFIFO) access registers

Only the core global, power and clock-gating, data FIFO access, and host port control and status registers can be accessed in both host and device modes. When the OTG\_FS controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG\_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

### 50.14.1 CSR memory map

The host and device mode registers occupy different addresses. All registers are implemented in the AHB clock domain.

#### Global CSR map

These registers are available in both host and device modes.

**Table 322. Core global control and status registers (CSRs)**

Acronym	Address offset	Register name
OTG_GOTGCTL	0x000	<a href="#">Section 50.15.1: OTG control and status register (OTG_GOTGCTL)</a>
OTG_GOTGINT	0x004	<a href="#">Section 50.15.2: OTG interrupt register (OTG_GOTGINT)</a>
OTG_GAHBCFG	0x008	<a href="#">Section 50.15.3: OTG AHB configuration register (OTG_GAHBCFG)</a>
OTG_GUSBCFG	0x00C	<a href="#">Section 50.15.4: OTG USB configuration register (OTG_GUSBCFG)</a>
OTG_GRSTCTL	0x010	<a href="#">Section 50.15.5: OTG reset register (OTG_GRSTCTL)</a>
OTG_GINTSTS	0x014	<a href="#">Section 50.15.6: OTG core interrupt register (OTG_GINTSTS)</a>
OTG_GINTMSK	0x018	<a href="#">Section 50.15.7: OTG interrupt mask register (OTG_GINTMSK)</a>

**Table 322. Core global control and status registers (CSRs) (continued)**

Acronym	Address offset	Register name
OTG_GRXSTSR	0x01C	<i>Section 50.15.8: OTG receive status debug read/OTG status read and pop registers (OTG_GRXSTSR/OTG_GRXSTSP)</i>
OTG_GRXSTSP	0x020	
OTG_GRXFSIZ	0x024	<i>Section 50.15.9: OTG receive FIFO size register (OTG_GRXFSIZ)</i>
OTG_HNPTXFSIZ/ OTG_DIEPTXF0 <sup>(1)</sup>	0x028	<i>Section 50.15.10: OTG host non-periodic transmit FIFO size register (OTG_HNPTXFSIZ)/Endpoint 0 Transmit FIFO size (OTG_DIEPTXF0)</i>
OTG_HNPTXSTS	0x02C	<i>Section 50.15.11: OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)</i>
OTG_GCCFG	0x038	<i>Section 50.15.12: OTG general core configuration register (OTG_GCCFG)</i>
OTG_CID	0x03C	<i>Section 50.15.13: OTG core ID register (OTG_CID)</i>
OTG_GLPMCFG	0x54	<i>Section 50.15.14: OTG core LPM configuration register (OTG_GLPMCFG)</i>
OTG_GPWRDN	0x058	<i>Section 50.15.15: OTG power down register (OTG_GPWRDN)</i>
OTG_GADPCTL	0x060	<i>Section 50.15.16: OTG ADP timer, control and status register (OTG_GADPCTL)</i>
OTG_HPTXFSIZ	0x100	<i>Section 50.15.17: OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)</i>
OTG_DIEPTFXx	0x104 0x108 ... 0x114	<i>Section 50.15.18: OTG device IN endpoint transmit FIFO size register (OTG_DIEPTFXx) (x = 1..5, where x is the FIFO number)</i>

1. The general rule is to use OTG\_HNPTXFSIZ for host mode and OTG\_DIEPTXF0 for device mode.

### Host-mode CSR map

These registers must be programmed every time the core changes to host mode.

**Table 323. Host-mode control and status registers (CSRs)**

Acronym	Offset address	Register name
OTG_HCFG	0x400	<i>Section 50.15.20: OTG host configuration register (OTG_HCFG)</i>
OTG_HFIR	0x404	<i>Section 50.15.21: OTG host frame interval register (OTG_HFIR)</i>
OTG_HFNUM	0x408	<i>Section 50.15.22: OTG host frame number/frame time remaining register (OTG_HFNUM)</i>
OTG_HPTXSTS	0x410	<i>Section 50.15.23: OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)</i>
OTG_HAINT	0x414	<i>Section 50.15.24: OTG host all channels interrupt register (OTG_HAINT)</i>
OTG_HAINTMSK	0x418	<i>Section 50.15.25: OTG host all channels interrupt mask register (OTG_HAINTMSK)</i>



Table 323. Host-mode control and status registers (CSRs) (continued)

Acronym	Offset address	Register name
OTG_HPRT	0x440	<a href="#">Section 50.15.26: OTG host port control and status register (OTG_HPRT)</a>
OTG_HCCHARx	0x500 0x520 ... 0x660	<a href="#">Section 50.15.27: OTG host channel x characteristics register (OTG_HCCHARx) (x = 0..11, where x = Channel number)</a>
OTG_HCINTx	0x508 0x528 .... 0x668	<a href="#">Section 50.15.28: OTG host channel x interrupt register (OTG_HCINTx) (x = 0..11, where x = Channel number)</a>
OTG_HCINTMSKx	0x50C 0x52C .... 0x66C	<a href="#">Section 50.15.29: OTG host channel x interrupt mask register (OTG_HCINTMSKx) (x = 0..11, where x = Channel number)</a>
OTG_HCTSIZx	0x510 0x530 .... 0x670	<a href="#">Section 50.15.30: OTG host channel x transfer size register (OTG_HCTSIZx) (x = 0..11, where x = Channel number)</a>

### Device-mode CSR map

These registers must be programmed every time the core changes to device mode.

Table 324. Device-mode control and status registers

Acronym	Offset address	Register name
OTG_DCFG	0x800	<a href="#">Section 50.15.32: OTG device configuration register (OTG_DCFG)</a>
OTG_DCTL	0x804	<a href="#">Section 50.15.33: OTG device control register (OTG_DCTL)</a>
OTG_DSTS	0x808	<a href="#">Section 50.15.34: OTG device status register (OTG_DSTS)</a>
OTG_DIEPMSK	0x810	<a href="#">Section 50.15.35: OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)</a>
OTG_DOEPMSK	0x814	<a href="#">Section 50.15.36: OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK)</a>
OTG_DAIN	0x818	<a href="#">Section 50.15.37: OTG device all endpoints interrupt register (OTG_DAIN)</a>
OTG_DAINMSK	0x81C	<a href="#">Section 50.15.38: OTG all endpoints interrupt mask register (OTG_DAINMSK)</a>
OTG_DVBUSDIS	0x828	<a href="#">Section 50.15.39: OTG device <math>V_{BUS}</math> discharge time register (OTG_DVBUSDIS)</a>
OTG_DVBUSPULSE	0x82C	<a href="#">Section 50.15.40: OTG device <math>V_{BUS}</math> pulsing time register (OTG_DVBUSPULSE)</a>

**Table 324. Device-mode control and status registers (continued)**

Acronym	Offset address	Register name
OTG_DIEPEMPMSK	0x834	<i>Section 50.15.41: OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK)</i>
OTG_DIEPCTL0	0x900	<i>Section 50.15.42: OTG device control IN endpoint 0 control register (OTG_DIEPCTL0)</i>
OTG_DIEPCTLx	0x920 0x940 ... 0x9A0	<i>Section 50.15.43: OTG device IN endpoint x control register (OTG_DIEPCTLx) (x = 1..5, where x = endpoint number)</i>
OTG_DIEPINTx	0x908 0x928 .... 0x988	<i>Section 50.15.44: OTG device IN endpoint x interrupt register (OTG_DIEPINTx) (x = 0..5, where x = Endpoint number)</i>
OTG_DIEPTSIZE0	0x910	<i>Section 50.15.45: OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZE0)</i>
OTG_DTXFSTSx	0x918 0x938 .... 0x998	<i>Section 50.15.46: OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx) (x = 0..5, where x = endpoint number)</i>
OTG_DIEPTSIZEx	0x930 0x950 ... 0x9B0	<i>Section 50.15.47: OTG device IN endpoint x transfer size register (OTG_DIEPTSIZEx) (x = 1..5, where x = endpoint number)</i>
OTG_DOEPCTL0	0xB00	<i>Section 50.15.48: OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)</i>
OTG_DOEPINTx	0xB08 0xB28 ... 0xBA8	<i>Section 50.15.49: OTG device OUT endpoint x interrupt register (OTG_DOEPINTx) (x = 0..5, where x = Endpoint number)</i>
OTG_DOEPTSIZE0	0xB10	<i>Section 50.15.50: OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZE0)</i>
OTG_DOEPCTLx	0xB20 0xB40 ... 0xBA0	<i>Section 50.15.51: OTG device OUT endpoint x control register (OTG_DOEPCTLx) (x = 1..5, where x = endpoint number)</i>
OTG_DOEPTSIZEx	0xB30 0xB50 ... 0xBB0	<i>Section 50.15.52: OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZEx) (x = 1..5, where x = Endpoint number)</i>

### Data FIFO (DFIFO) access register map

These registers, available in both host and device modes, are used to read or write the FIFO space for a specific endpoint or a channel, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel.

**Table 325. Data FIFO (DFIFO) access register map**

FIFO access register section	Offset address	Access
Device IN endpoint 0/Host OUT Channel 0: DFIFO write access Device OUT endpoint 0/Host IN Channel 0: DFIFO read access	0x1000–0x1FFC	w r
Device IN endpoint 1/Host OUT Channel 1: DFIFO write access Device OUT endpoint 1/Host IN Channel 1: DFIFO read access	0x2000–0x2FFC	w r
...	...	...
Device IN endpoint x <sup>(1)</sup> /Host OUT Channel x <sup>(1)</sup> : DFIFO write access Device OUT endpoint x <sup>(1)</sup> /Host IN Channel x <sup>(1)</sup> : DFIFO read access	0xX000–0xXFFC	w r

1. Where x is 5 in device mode and 11 in host mode.

### Power and clock gating CSR map

There is a single register for power and clock gating. It is available in both host and device modes.

**Table 326. Power and clock gating control and status registers**

Acronym	Offset address	Register name
OTG_PCGCCTL	0xE00–0xE04	<a href="#">Section 50.15.53: OTG power and clock gating control register (OTG_PCGCCTL)</a>

## 50.15 OTG\_FS registers

These registers are available in both host and device modes, and do not need to be reprogrammed when switching between these modes.

Bit values in the register descriptions are expressed in binary unless otherwise specified.

### 50.15.1 OTG control and status register (OTG\_GOTGCTL)

Address offset: 0x000

Reset value: 0x0001 0000

The OTG\_GOTGCTL register controls the behavior and reflects the status of the OTG function of the core.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CUR MOD	OTG VER	BSVLD	ASVLD	DBCT	CID STS
										r	rw	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EHEN	DHNP EN	HSHNP EN	HNP RQ	HNG SCS	BVALO VAL	BVALO EN	AVALO VAL	AVALO EN	VBVAL OVAL	VBVAL OEN	SRQ	SRQ SCS
			rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CURMOD**: Current mode of operation  
 Indicates the current mode (host or device).  
 0: Device mode  
 1: Host mode

Bit 20 **OTGVER**: OTG version  
 Selects the OTG revision.  
 0:OTG Version 1.3. OTG1.3 is obsolete for new product development.  
 1:OTG Version 2.0. In this version the core supports only data line pulsing for SRP.

Bit 19 **BSVLD**: B-session valid  
 Indicates the device mode transceiver status.  
 0: B-session is not valid.  
 1: B-session is valid.  
 In OTG mode, the user can use this bit to determine if the device is connected or disconnected.  
*Note: Only accessible in device mode.*

Bit 18 **ASVLD**: A-session valid  
 Indicates the host mode transceiver status.  
 0: A-session is not valid  
 1: A-session is valid  
*Note: Only accessible in host mode.*

Bit 17 **DBCT**: Long/short debounce time  
 Indicates the debounce time of a detected connection.  
 0: Long debounce time, used for physical connections (100 ms + 2.5 μs)  
 1: Short debounce time, used for soft connections (2.5 μs)  
*Note: Only accessible in host mode.*

Bit 16 **CIDSTS**: Connector ID status  
 Indicates the connector ID status on a connect event.  
 0: The OTG\_FS controller is in A-device mode  
 1: The OTG\_FS controller is in B-device mode  
*Note: Accessible in both device and host modes.*

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **EHEN**: Embedded host enable

It is used to select between OTG A device state machine and embedded host state machine.

0: OTG A device state machine is selected

1: Embedded host state machine is selected

Bit 11 **DHNPEN**: Device HNP enabled

The application sets this bit when it successfully receives a SetFeature.SetHNPEnable command from the connected USB host.

0: HNP is not enabled in the application

1: HNP is enabled in the application

*Note: Only accessible in device mode.*

Bit 10 **HSHNPEN**: host set HNP enable

The application sets this bit when it has successfully enabled HNP (using the SetFeature.SetHNPEnable command) on the connected device.

0: Host Set HNP is not enabled

1: Host Set HNP is enabled

*Note: Only accessible in host mode.*

Bit 9 **HNPRQ**: HNP request

The application sets this bit to initiate an HNP request to the connected USB host. The application can clear this bit by writing a 0 when the host negotiation success status change bit in the OTG\_GOTGINT register (HNSSCHG bit in OTG\_GOTGINT) is set. The core clears this bit when the HNSSCHG bit is cleared.

0: No HNP request

1: HNP request

*Note: Only accessible in device mode.*

Bit 8 **HNGSCS**: Host negotiation success

The core sets this bit when host negotiation is successful. The core clears this bit when the HNP request (HNPRQ) bit in this register is set.

0: Host negotiation failure

1: Host negotiation success

*Note: Only accessible in device mode.*

Bit 7 **BVALOVAL**: B-peripheral session valid override value.

This bit is used to set override value for Bvalid signal when BVALOEN bit is set.

0: Bvalid value is '0' when BVALOEN = 1

1: Bvalid value is '1' when BVALOEN = 1

*Note: Only accessible in device mode.*

Bit 6 **BVALOEN**: B-peripheral session valid override enable.

This bit is used to enable/disable the software to override the Bvalid signal using the BVALOVAL bit.

0: Override is disabled and Bvalid signal from the respective PHY selected is used internally by the core

1: Internally Bvalid received from the PHY is overridden with BVALOVAL bit value

*Note: Only accessible in device mode.*

Bit 5 **AVALOVAL**: A-peripheral session valid override value.

This bit is used to set override value for Avalid signal when AVALOEN bit is set.

0: Avalid value is '0' when AVALOEN = 1

1: Avalid value is '1' when AVALOEN = 1

*Note: Only accessible in host mode.*

- Bit 4 **AVALOEN**: A-peripheral session valid override enable.  
This bit is used to enable/disable the software to override the Avalid signal using the AVALOVAL bit.  
0: Override is disabled and Avalid signal from the respective PHY selected is used internally by the core  
1: Internally Avalid received from the PHY is overridden with AVALOVAL bit value  
*Note: Only accessible in host mode.*
- Bit 3 **VBVALOVAL**:  $V_{BUS}$  valid override value.  
This bit is used to set override value for vbusvalid signal when VBVALOEN bit is set.  
0: vbusvalid value is '0' when VBVALOEN = 1  
1: vbusvalid value is '1' when VBVALOEN = 1  
*Note: Only accessible in host mode.*
- Bit 2 **VBVALOEN**:  $V_{BUS}$  valid override enable.  
This bit is used to enable/disable the software to override the vbusvalid signal using the VBVALOVAL bit.  
0: Override is disabled and vbusvalid signal from the respective PHY selected is used internally by the core  
1: Internally vbusvalid received from the PHY is overridden with VBVALOVAL bit value  
*Note: Only accessible in host mode.*
- Bit 1 **SRQ**: Session request  
The application sets this bit to initiate a session request on the USB. The application can clear this bit by writing a 0 when the host negotiation success status change bit in the OTG\_GOTGINT register (HNSSCHG bit in OTG\_GOTGINT) is set. The core clears this bit when the HNSSCHG bit is cleared.  
If the user uses the USB 1.1 full-speed serial transceiver interface to initiate the session request, the application must wait until  $V_{BUS}$  discharges to 0.2 V, after the B-session valid bit in this register (BSVLD bit in OTG\_GOTGCTL) is cleared.  
0: No session request  
1: Session request  
*Note: Only accessible in device mode.*
- Bit 0 **SRQSCS**: Session request success  
The core sets this bit when a session request initiation is successful.  
0: Session request failure  
1: Session request success  
*Note: Only accessible in device mode.*

### 50.15.2 OTG interrupt register (OTG\_GOTGINT)

Address offset: 0x04

Reset value: 0x0000 0000

The application reads this register whenever there is an OTG interrupt and clears the bits in this register to clear the OTG interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID CHNG	DBC DNE	ADTO CHG	HNG DET	Res.
											rc_w1	rc_w1	rc_w1	rc_w1	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	HNSS CHG	SRSS CHG	Res.	Res.	Res.	Res.	Res.	SEDET	Res.	Res.
						rc_w1	rc_w1						rc_w1		

Bits 31:21 Reserved, must be kept at reset value.

**Bit 20 IDCHNG:**

This bit when set indicates that there is a change in the value of the ID input pin.

**Bit 19 DBCDNE:** Debounce done

The core sets this bit when the debounce is completed after the device connect. The application can start driving USB reset after seeing this interrupt. This bit is only valid when the HNP Capable or SRP Capable bit is set in the OTG\_GUSBCFG register (HNPCAP bit or SRPCAP bit in OTG\_GUSBCFG, respectively).

*Note: Only accessible in host mode.*

**Bit 18 ADTOCHG:** A-device timeout change

The core sets this bit to indicate that the A-device has timed out while waiting for the B-device to connect.

*Note: Accessible in both device and host modes.*

**Bit 17 HNGDET:** Host negotiation detected

The core sets this bit when it detects a host negotiation request on the USB.

*Note: Accessible in both device and host modes.*

Bits 16:10 Reserved, must be kept at reset value.

**Bit 9 HNSSCHG:** Host negotiation success status change

The core sets this bit on the success or failure of a USB host negotiation request. The application must read the host negotiation success bit of the OTG\_GOTGCTL register (HNGSCS bit in OTG\_GOTGCTL) to check for success or failure.

*Note: Accessible in both device and host modes.*

Bits 7:3 Reserved, must be kept at reset value.

Bit 8 **SRSSCHG**: Session request success status change

The core sets this bit on the success or failure of a session request. The application must read the session request success bit in the OTG\_GOTGCTL register (SRQSCS bit in OTG\_GOTGCTL) to check for success or failure.

*Note: Accessible in both device and host modes.*

Bit 2 **SEDET**: Session end detected

The core sets this bit to indicate that the level of the voltage on  $V_{BUS}$  is no longer valid for a B-Peripheral session when  $V_{BUS} < 0.8$  V.

*Note: Accessible in both device and host modes.*

Bits 1:0 Reserved, must be kept at reset value.

### 50.15.3 OTG AHB configuration register (OTG\_GAHBCFG)

Address offset: 0x008

Reset value: 0x0000 0000

This register can be used to configure the core after power-on or a change in mode. This register mainly contains AHB system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	PTXFE LVL	TXFE LVL	Res.	Res.	Res.	Res.	Res.	Res.	GINT MSK
							rw	rw							rw



Bits 31:9 Reserved, must be kept at reset value.

**Bit 8 PTXFELVL:** Periodic Tx FIFO empty level

Indicates when the periodic Tx FIFO empty interrupt bit in the OTG\_GINTSTS register (PTXFE bit in OTG\_GINTSTS) is triggered.

0: PTXFE (in OTG\_GINTSTS) interrupt indicates that the Periodic Tx FIFO is half empty

1: PTXFE (in OTG\_GINTSTS) interrupt indicates that the Periodic Tx FIFO is completely empty

*Note: Only accessible in host mode.*

**Bit 7 TXFELVL:** Tx FIFO empty level

In device mode, this bit indicates when IN endpoint Transmit FIFO empty interrupt (TXFE in OTG\_DIEPINTx) is triggered:

0: The TXFE (in OTG\_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is half empty

1: The TXFE (in OTG\_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is completely empty

In host mode, this bit indicates when the nonperiodic Tx FIFO empty interrupt (NPTXFE bit in OTG\_GINTSTS) is triggered:

0: The NPTXFE (in OTG\_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is half empty

1: The NPTXFE (in OTG\_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is completely empty

Bits 6:1 Reserved, must be kept at reset value.

**Bit 0 GINTMSK:** Global interrupt mask

The application uses this bit to mask or unmask the interrupt line assertion to itself.

Irrespective of this bit's setting, the interrupt status registers are updated by the core.

0: Mask the interrupt assertion to the application.

1: Unmask the interrupt assertion to the application.

*Note: Accessible in both device and host modes.*

### 50.15.4 OTG USB configuration register (OTG\_GUSBCFG)

Address offset: 0x00C

Reset value: 0x0000 1440

This register can be used to configure the core after power-on or a changing to host mode or device mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the AHB or the USB. Do not make changes to this register after the initial programming.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	FD MOD	FH MOD	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.	TRDT				HNP CAP	SRP CAP	Res.	PHY SEL	Res.	Res.	Res.	TOTAL		
		rw				rw	rw		r				rw		



- Bit 31 Reserved, must be kept at reset value.
- Bit 30 **FDMOD**: Force device mode  
Writing a 1 to this bit, forces the core to device mode irrespective of the OTG\_ID input pin.  
0: Normal mode  
1: Force device mode  
After setting the force bit, the application must wait at least 25 ms before the change takes effect.  
*Note: Accessible in both device and host modes.*
- Bit 29 **FHMOD**: Force host mode  
Writing a 1 to this bit, forces the core to host mode irrespective of the OTG\_ID input pin.  
0: Normal mode  
1: Force host mode  
After setting the force bit, the application must wait at least 25 ms before the change takes effect.  
*Note: Accessible in both device and host modes.*
- Bits 28:26 Reserved, must be kept at reset value.
- Bits 25:23 Reserved, must be kept at reset value.
- Bit 22 Reserved, must be kept at reset value.
- Bits 21:16 Reserved, must be kept at reset value.
- Bit 15 Reserved, must be kept at reset value.
- Bit 14 Reserved, must be kept at reset value.
- Bits 13:10 **TRDT[3:0]**: USB turnaround time  
These bits allows to set the turnaround time in PHY clocks. They must be configured according to [Table 327: TRDT values \(FS\)](#), depending on the application AHB frequency. Higher TRDT values allow stretching the USB response time to IN tokens in order to compensate for longer AHB read access latency to the data FIFO.  
*Note: Only accessible in device mode.*
- Bit 9 **HNPCAP**: HNP-capable  
The application uses this bit to control the OTG\_FS controller's HNP capabilities.  
0: HNP capability is not enabled.  
1: HNP capability is enabled.  
*Note: Accessible in both device and host modes.*
- Bit 8 **SRPCAP**: SRP-capable  
The application uses this bit to control the OTG\_FS controller's SRP capabilities. If the core operates as a non-SRP-capable B-device, it cannot request the connected A-device (host) to activate  $V_{BUS}$  and start a session.  
0: SRP capability is not enabled.  
1: SRP capability is enabled.  
*Note: Accessible in both device and host modes.*
- Bit 7 Reserved, must be kept at reset value.
- Bit 6 **PHYSEL**: Full Speed serial transceiver select  
This bit is always 1 with read-only access.
- Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **TOCAL[2:0]**: FS timeout calibration

The number of PHY clocks that the application programs in this field is added to the full-speed interpacket timeout duration in the core to account for any additional delays introduced by the PHY. This can be required, because the delay introduced by the PHY in generating the line state condition can vary from one PHY to another.

The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock is 0.25 bit times.

**Table 327. TRDT values (FS)**

AHB frequency range (MHz)		TRDT minimum value
Min	Max	
14.2	15	0xF
15	16	0xE
16	17.2	0xD
17.2	18.5	0xC
18.5	20	0xB
20	21.8	0xA
21.8	24	0x9
24	27.5	0x8
27.5	32	0x7
32	-	0x6

### 50.15.5 OTG reset register (OTG\_GRSTCTL)

Address offset: 0x10

Reset value: 0x8000 0000

The application uses this register to reset various hardware features inside the core.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHB IDL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TXFNUM					TXF FLSH	RXF FLSH	Res.	FCRST	PSRST	CSRST
					rw					rs	rs		rs	rs	r

Bit 31 **AHBIDL**: AHB master idle

Indicates that the AHB master state machine is in the Idle condition.

*Note: Accessible in both device and host modes.*

Bits 30:11 Reserved, must be kept at reset value.

Bits 10:6 **TXFNUM[4:0]**: Tx FIFO number

This is the FIFO number that must be flushed using the Tx FIFO Flush bit. This field must not be changed until the core clears the Tx FIFO Flush bit.

00000:

- Non-periodic Tx FIFO flush in host mode
- Tx FIFO 0 flush in device mode

00001:

- Periodic Tx FIFO flush in host mode
- Tx FIFO 1 flush in device mode

00010: Tx FIFO 2 flush in device mode

...

01111: Tx FIFO 15 flush in device mode

10000: Flush all the transmit FIFOs in device or host mode.

*Note: Accessible in both device and host modes.*

Bit 5 **TXFFLSH**: Tx FIFO flush

This bit selectively flushes a single or all transmit FIFOs, but cannot do so if the core is in the midst of a transaction.

The application must write this bit only after checking that the core is neither writing to the Tx FIFO nor reading from the Tx FIFO. Verify using these registers:

Read—NAK Effective interrupt ensures the core is not reading from the FIFO

Write—AHBIDL bit in OTG\_GRSTCTL ensures the core is not writing anything to the FIFO.

Flushing is normally recommended when FIFOs are reconfigured. FIFO flushing is also recommended during device endpoint disable. The application must wait until the core clears this bit before performing any operations. This bit takes eight clocks to clear, using the slower clock of phy\_clk or hclk.

*Note: Accessible in both device and host modes.*

Bit 4 **RXFFLSH**: Rx FIFO flush

The application can flush the entire Rx FIFO using this bit, but must first ensure that the core is not in the middle of a transaction.

The application must only write to this bit after checking that the core is neither reading from the Rx FIFO nor writing to the Rx FIFO.

The application must wait until the bit is cleared before performing any other operations. This bit requires 8 clocks (slowest of PHY or AHB clock) to clear.

*Note: Accessible in both device and host modes.*

Bit 3 Reserved, must be kept at reset value.

**Bit 2 FCRST:** Host frame counter reset

The application writes this bit to reset the frame number counter inside the core. When the frame counter is reset, the subsequent SOF sent out by the core has a frame number of 0. When application writes '1' to the bit, it might not be able to read back the value as it will get cleared by the core in a few clock cycles.

*Note:* Only accessible in host mode.

**Bit 1 PSRST:** Partial soft reset

Resets the internal state machines but keeps the enumeration info. Could be used to recover some specific PHY errors.

*Note:* Accessible in both device and host modes.

**Bit 0 CSRST:** Core soft reset

Resets the HCLK and PHY clock domains as follows:

Clears the interrupts and all the CSR register bits except for the following bits:

- GATEHCLK bit in OTG\_PCGCCTL
- STPPCLK bit in OTG\_PCGCCTL
- FLSPPCS bits in OTG\_HCFG
- DSPD bit in OTG\_DCFG
- SDIS bit in OTG\_DCTL
- OTG\_GCCFG register
- OTG\_GPWRDN register
- OTG\_GADPCTL register

All module state machines (except for the AHB slave unit) are reset to the Idle state, and all the transmit FIFOs and the receive FIFO are flushed.

Any transactions on the AHB Master are terminated as soon as possible, after completing the last data phase of an AHB transfer. Any transactions on the USB are terminated immediately. When ADP feature is enabled, the power management module is not reset by the core soft reset.

The application can write to this bit any time it wants to reset the core. This is a self-clearing bit and the core clears this bit after all the necessary logic is reset in the core, which can take several clocks, depending on the current state of the core. Once this bit has been cleared, the software must wait at least 3 PHY clocks before accessing the PHY domain (synchronization delay). The software must also check that bit 31 in this register is set to 1 (AHB Master is Idle) before starting any operation.

Typically, the software reset is used during software development and also when the user dynamically changes the PHY selection bits in the above listed USB configuration registers. When you change the PHY, the corresponding clock for the PHY is selected and used in the PHY domain. Once a new clock is selected, the PHY domain has to be reset for proper operation.

*Note:* Accessible in both device and host modes.

### 50.15.6 OTG core interrupt register (OTG\_GINTSTS)

Address offset: 0x014

Reset value: 0x1400 0020

This register interrupts the application for system-level events in the current mode (device mode or host mode).

Some of the bits in this register are valid only in host mode, while others are valid in device mode only. This register also indicates the current mode. To clear the interrupt status bits of the rc\_w1 type, the application must write 1 into the bit.

The FIFO status interrupts are read-only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the OTG\_GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WKUP INT	SRQ INT	DISC INT	CIDS CHG	LPM INT	PTXFE	HCINT	HPRT INT	RST DET	Res.	IPXFR/ IN COMP ISO OUT	IISOI XFR	OEP INT	IEPINT	Res.	Res.
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	r	r	r	rc_w1		rc_w1	rc_w1	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPF	ISOO DRP	ENUM DNE	USB RST	USB SUSP	ESUSP	Res.	Res.	GO NAK EFF	GI NAK EFF	NPTXFE	RXF LVL	SOF	OTG INT	MMIS	CMOD
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1			r	r	r	r	rc_w1	r	rc_w1	r

**Bit 31 WKUPINT:** Resume/remote wakeup detected interrupt  
 Wakeup interrupt during suspend(L2) or LPM(L1) state.  
 – During suspend(L2):  
 In device mode, this interrupt is asserted when a resume is detected on the USB. In host mode, this interrupt is asserted when a remote wakeup is detected on the USB.  
 – During LPM(L1):  
 This interrupt is asserted for either host initiated resume or device initiated remote wakeup on USB.

*Note: Accessible in both device and host modes.*

**Bit 30 SRQINT:** Session request/new session detected interrupt  
 In host mode, this interrupt is asserted when a session request is detected from the device. In device mode, this interrupt is asserted when V<sub>BUS</sub> is in the valid range for a B-peripheral device. Accessible in both device and host modes.

**Bit 29 DISCINT:** Disconnect detected interrupt  
 Asserted when a device disconnect is detected.  
*Note: Only accessible in host mode.*

**Bit 28 CIDSCHG:** Connector ID status change  
 The core sets this bit when there is a change in connector ID status.  
*Note: Accessible in both device and host modes.*



**Bit 27 LPMINT:** LPM interrupt

In device mode, this interrupt is asserted when the device receives an LPM transaction and responds with a non-ERRORed response.

In host mode, this interrupt is asserted when the device responds to an LPM transaction with a non-ERRORed response or when the host core has completed LPM transactions for the programmed number of times (RETRYCNT bit in OTG\_GLPMCFG).

This field is valid only if the LPMEN bit in OTG\_GLPMCFG is set to 1.

**Bit 26 PTXFE:** Periodic Tx FIFO empty

Asserted when the periodic transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the periodic request queue. The half or completely empty status is determined by the periodic Tx FIFO empty level bit in the OTG\_GAHBCFG register (PTXFELVL bit in OTG\_GAHBCFG).

*Note: Only accessible in host mode.*

**Bit 25 HCINT:** Host channels interrupt

The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in host mode). The application must read the OTG\_HAINT register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding OTG\_HCINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the OTG\_HCINTx register to clear this bit.

*Note: Only accessible in host mode.*

**Bit 24 HPRTINT:** Host port interrupt

The core sets this bit to indicate a change in port status of one of the OTG\_FS controller ports in host mode. The application must read the OTG\_HPRT register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG\_HPRT register to clear this bit.

*Note: Only accessible in host mode.*

**Bit 23 RSTDET:** Reset detected interrupt

In device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in suspend.

*Note: Only accessible in device mode.*

**Bit 22** Reserved, must be kept at reset value .**Bit 21 IPXFR:** Incomplete periodic transfer

In host mode, the core sets this interrupt bit when there are incomplete periodic transactions still pending, which are scheduled for the current frame.

**INCOMPISOOUT:** Incomplete isochronous OUT transfer

In device mode, the core sets this interrupt to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

**Bit 20 IISOIFR:** Incomplete isochronous IN transfer

The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

*Note: Only accessible in device mode.*

- Bit 19 **OEPINT**: OUT endpoint interrupt  
The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in device mode). The application must read the OTG\_DAIN register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding OTG\_DOEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG\_DOEPINTx register to clear this bit.  
*Note: Only accessible in device mode.*
- Bit 18 **IIEPINT**: IN endpoint interrupt  
The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in device mode). The application must read the OTG\_DAIN register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding OTG\_DIEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG\_DIEPINTx register to clear this bit.  
*Note: Only accessible in device mode.*
- Bits 17:16 Reserved, must be kept at reset value.
- Bit 15 **EOPF**: End of periodic frame interrupt  
Indicates that the period specified in the periodic frame interval field of the OTG\_DCFG register (PFIVL bit in OTG\_DCFG) has been reached in the current frame.  
*Note: Only accessible in device mode.*
- Bit 14 **ISOODRP**: Isochronous OUT packet dropped interrupt  
The core sets this bit when it fails to write an isochronous OUT packet into the Rx FIFO because the Rx FIFO does not have enough space to accommodate a maximum size packet for the isochronous OUT endpoint.  
*Note: Only accessible in device mode.*
- Bit 13 **ENUMDNE**: Enumeration done  
The core sets this bit to indicate that speed enumeration is complete. The application must read the OTG\_DSTS register to obtain the enumerated speed.  
*Note: Only accessible in device mode.*
- Bit 12 **USBRST**: USB reset  
The core sets this bit to indicate that a reset is detected on the USB.  
*Note: Only accessible in device mode.*
- Bit 11 **USBSUSP**: USB suspend  
The core sets this bit to indicate that a suspend was detected on the USB. The core enters the suspended state when there is no activity on the data lines for an extended period of time.  
*Note: Only accessible in device mode.*
- Bit 10 **ESUSP**: Early suspend  
The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.  
*Note: Only accessible in device mode.*
- Bits 9:8 Reserved, must be kept at reset value.
- Bit 7 **GONAKEFF**: Global OUT NAK effective  
Indicates that the Set global OUT NAK bit in the OTG\_DCTL register (SGONAK bit in OTG\_DCTL), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear global OUT NAK bit in the OTG\_DCTL register (CGONAK bit in OTG\_DCTL).  
*Note: Only accessible in device mode.*



- Bit 6 **GINAKEFF**: Global IN non-periodic NAK effective  
 Indicates that the Set global non-periodic IN NAK bit in the OTG\_DCTL register (SGINAK bit in OTG\_DCTL), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear global non-periodic IN NAK bit in the OTG\_DCTL register (CGINAK bit in OTG\_DCTL).  
 This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.  
*Note: Only accessible in device mode.*
- Bit 5 **NPTXFE**: Non-periodic Tx FIFO empty  
 This interrupt is asserted when the non-periodic Tx FIFO is either half or completely empty, and there is space for at least one entry to be written to the non-periodic transmit request queue. The half or completely empty status is determined by the non-periodic Tx FIFO empty level bit in the OTG\_GAHBCFG register (TXFELVL bit in OTG\_GAHBCFG).  
*Note: Accessible in host mode only.*
- Bit 4 **RXFLVL**: Rx FIFO non-empty  
 Indicates that there is at least one packet pending to be read from the Rx FIFO.  
*Note: Accessible in both host and device modes.*
- Bit 3 **SOF**: Start of frame  
 In host mode, the core sets this bit to indicate that an SOF (FS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt.  
 In device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the OTG\_DSTS register to get the current frame number. This interrupt is seen only when the core is operating in FS.  
*Note: This register may return '1' if read immediately after power on reset. If the register bit reads '1' immediately after power on reset it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.*  
*Note: Accessible in both host and device modes.*
- Bit 2 **OTGINT**: OTG interrupt  
 The core sets this bit to indicate an OTG protocol event. The application must read the OTG interrupt status (OTG\_GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG\_GOTGINT register to clear this bit.  
*Note: Accessible in both host and device modes.*
- Bit 1 **MMIS**: Mode mismatch interrupt  
 The core sets this bit when the application is trying to access:  
 – A host mode register, when the core is operating in device mode  
 – A device mode register, when the core is operating in host mode  
 The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.  
*Note: Accessible in both host and device modes.*
- Bit 0 **CMOD**: Current mode of operation  
 Indicates the current mode.  
 0: Device mode  
 1: Host mode  
*Note: Accessible in both host and device modes.*

### 50.15.7 OTG interrupt mask register (OTG\_GINTMSK)

Address offset: 0x018

Reset value: 0x0000 0000

This register works with the core interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the core interrupt (OTG\_GINTSTS) register bit corresponding to that interrupt is still set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUIM	SRQIM	DISCINT	CIDSCHGM	LPMINTM	PTXFEM	HCIM	PRTIM	RSTDETM	Res.	IPXFRM/IISOXFRM	IISOIXFRM	OEPINT	IEPINT	Res.	Res.
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
EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Res.	Res.	GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	

Bit 31 **WUIM**: Resume/remote wakeup detected interrupt mask

- 0: Masked interrupt
- 1: Unmasked interrupt

*Note: Accessible in both host and device modes.*

Bit 30 **SRQIM**: Session request/new session detected interrupt mask

- 0: Masked interrupt
- 1: Unmasked interrupt

*Note: Accessible in both host and device modes.*

Bit 29 **DISCINT**: Disconnect detected interrupt mask

- 0: Masked interrupt
- 1: Unmasked interrupt

*Note: Only accessible in host mode.*

Bit 28 **CIDSCHGM**: Connector ID status change mask

- 0: Masked interrupt
- 1: Unmasked interrupt

*Note: Accessible in both host and device modes.*

Bit 27 **LPMINTM**: LPM interrupt mask

- 0: Masked interrupt
- 1: Unmasked interrupt

*Note: Accessible in both host and device modes.*

Bit 26 **PTXFEM**: Periodic Tx FIFO empty mask

- 0: Masked interrupt
- 1: Unmasked interrupt

*Note: Only accessible in host mode.*

Bit 25 **HCIM**: Host channels interrupt mask

- 0: Masked interrupt
- 1: Unmasked interrupt

*Note: Only accessible in host mode.*

- Bit 24 **PRTIM**: Host port interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in host mode.*
- Bit 23 **RSTDETM**: Reset detected interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in device mode.*
- Bit 22 Reserved, must be kept at reset value .
- Bit 21 **IPXFRM**: Incomplete periodic transfer mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in host mode.*  
**IISOXFRM**: Incomplete isochronous OUT transfer mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in device mode.*
- Bit 20 **IISOIXFRM**: Incomplete isochronous IN transfer mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in device mode.*
- Bit 19 **OEPIINT**: OUT endpoints interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in device mode.*
- Bit 18 **IEPIINT**: IN endpoints interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in device mode.*
- Bits 17:16 Reserved, must be kept at reset value.
- Bit 15 **EOPFM**: End of periodic frame interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in device mode.*
- Bit 14 **ISOODRPM**: Isochronous OUT packet dropped interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in device mode.*
- Bit 13 **ENUMDNEM**: Enumeration done mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in device mode.*

- Bit 12 **USBRST**: USB reset mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in device mode.*
- Bit 11 **USBSUSPM**: USB suspend mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in device mode.*
- Bit 10 **ESUSPM**: Early suspend mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in device mode.*
- Bits 9:8 Reserved, must be kept at reset value.
- Bit 7 **GONAKEFFM**: Global OUT NAK effective mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in device mode.*
- Bit 6 **GINAKEFFM**: Global non-periodic IN NAK effective mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in device mode.*
- Bit 5 **NPTXFEM**: Non-periodic Tx FIFO empty mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in host mode.*
- Bit 4 **RXFLVLM**: Receive FIFO non-empty mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Accessible in both device and host modes.*
- Bit 3 **SOFM**: Start of frame mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Accessible in both device and host modes.*
- Bit 2 **OTGINT**: OTG interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Accessible in both device and host modes.*
- Bit 1 **MMISM**: Mode mismatch interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Accessible in both device and host modes.*
- Bit 0 Reserved, must be kept at reset value.

### 50.15.8 OTG receive status debug read/OTG status read and pop registers (OTG\_GRXSTSR/OTG\_GRXSTSP)

Address offset for read: 0x01C

Address offset for pop: 0x020

Reset value: 0x0000 0000

A read to the receive status debug read register returns the contents of the top of the receive FIFO. A read to the receive status read and pop register additionally pops the top data entry out of the Rx FIFO.

The receive status contents must be interpreted differently in host and device modes. The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG\_GINTSTS) is asserted.

#### Host 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.	PKTSTS[3:0]				DPID
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID	BCNT[10:0]										CHNUM[3:0]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **PKTSTS[3:0]**: Packet status

- Indicates the status of the received packet
- 0010: IN data packet received
- 0011: IN transfer completed (triggers an interrupt)
- 0101: Data toggle error (triggers an interrupt)
- 0111: Channel halted (triggers an interrupt)
- Others: Reserved

Bits 16:15 **DPID**: Data PID

- Indicates the data PID of the received packet
- 00: DATA0
- 10: DATA1

Bits 14:4 **BCNT[10:0]**: Byte count

Indicates the byte count of the received IN data packet.

Bits 3:0 **CHNUM[3:0]**: Channel number

Indicates the channel number to which the current received packet belongs.

**Device mode:**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	STSPH ST	Res.	Res.	FRMNUM[3:0]				PKTSTS[3:0]				DPID[1]
				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
DPID[0]	BCNT[10:0]										EPNUM[3:0]				
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 **STSPHST**: Status phase start

Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern.

Bits 26:25 Reserved, must be kept at reset value.

Bits 24:21 **FRMNUM[3:0]**: Frame number

This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Bits 20:17 **PKTSTS[3:0]**: Packet status

Indicates the status of the received packet  
 0001: Global OUT NAK (triggers an interrupt)  
 0010: OUT data packet received  
 0011: OUT transfer completed (triggers an interrupt)  
 0100: SETUP transaction completed (triggers an interrupt)  
 0110: SETUP data packet received  
 Others: Reserved

Bits 16:15 **DPID[1:0]**: Data PID

Indicates the data PID of the received OUT data packet  
 00: DATA0  
 10: DATA1

Bits 14:4 **BCNT[10:0]**: Byte count

Indicates the byte count of the received data packet.

Bits 3:0 **EPNUM[3:0]**: Endpoint number

Indicates the endpoint number to which the current received packet belongs.

### 50.15.9 OTG receive FIFO size register (OTG\_GRXFSIZ)

Address offset: 0x024

Reset value: 0x0000 0200

The application can program the RAM size that must be allocated to the Rx FIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFD[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 **RXFD[15:0]**: Rx FIFO depth

This value is in terms of 32-bit words.

Minimum value is 16

Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

### 50.15.10 OTG host non-periodic transmit FIFO size register (OTG\_HNPTXFSIZ)/Endpoint 0 Transmit FIFO size (OTG\_DIEPTXF0)

Address offset: 0x028

Reset value: 0x0200 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPTXFD/TX0FD[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
NPTXFSA/TX0FSA[15:0]															
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

#### Host mode

Bits 31:16 **NPTXFD[15:0]**: Non-periodic Tx FIFO depth

This value is in terms of 32-bit words.

Minimum value is 16

Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **NPTXFSA[15:0]**: Non-periodic transmit RAM start address

This field configures the memory start address for non-periodic transmit FIFO RAM.

**Device mode**

Bits 31:16 **TX0FD**: Endpoint 0 Tx FIFO depth

This value is in terms of 32-bit words.

Minimum value is 16

Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **TX0FSA**: Endpoint 0 transmit RAM start address

This field configures the memory start address for the endpoint 0 transmit FIFO RAM.

**50.15.11 OTG non-periodic transmit FIFO/queue status register (OTG\_HNPTXSTS)**

Address offset: 0x02C

Reset value: 0x0008 0200

*Note:* In device mode, this register is not valid.

This read-only register contains the free space information for the non-periodic Tx FIFO and the non-periodic transmit request queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	NPTXQTOP[6:0]							NPTQXSAV[7:0]							
	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
NPTXFSAV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r



Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **NPTXQTOP[6:0]**: Top of the non-periodic transmit request queue  
 Entry in the non-periodic Tx request queue that is currently being processed by the MAC.  
 Bits 30:27: Channel/endpoint number  
 Bits 26:25:  
 00: IN/OUT token  
 01: Zero-length transmit packet (device IN/host OUT)  
 11: Channel halt command  
 Bit 24: Terminate (last entry for selected channel/endpoint)

Bits 23:16 **NPTQXSAV[7:0]**: Non-periodic transmit request queue space available  
 Indicates the amount of free space available in the non-periodic transmit request queue.  
 This queue holds both IN and OUT requests.  
 0: Non-periodic transmit request queue is full  
 1: 1 location available  
 2: locations available  
 n: n locations available (0 ≤ n ≤ 8)  
 Others: Reserved

Bits 15:0 **NPTXFSAV[15:0]**: Non-periodic Tx FIFO space available  
 Indicates the amount of free space available in the non-periodic Tx FIFO.  
 Values are in terms of 32-bit words.  
 0: Non-periodic Tx FIFO is full  
 1: 1 word available  
 2: 2 words available  
 n: n words available (where 0 ≤ n ≤ 512)  
 Others: Reserved

### 50.15.12 OTG general core configuration register (OTG\_GCCFG)

Address offset: 0x038

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBDEN	SDEN	PDEN	DCDEN	BCDEN	PWRDWN
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2DET	SDET	PDET	DCDET
												r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **VBDEN**: USB  $V_{BUS}$  detection enable

Enables  $V_{BUS}$  sensing comparators to detect  $V_{BUS}$  valid levels on the  $V_{BUS}$  PAD for USB host and device operation. If HNP and/or SRP support is enabled,  $V_{BUS}$  comparators are automatically enabled independently of VBDEN value.

0 =  $V_{BUS}$  detection disabled

1 =  $V_{BUS}$  detection enabled

Bit 20 **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 19 **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 18 **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 17 **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.

Bit 16 **PWRDWN**: Power down control

Used to activate the transceiver in transmission/reception. When reset, the transceiver is kept in power-down. When set, the BCD function must be off (BCDEN=0).

0 = USB FS transceiver disabled

1 = USB FS transceiver enabled

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **PS2DET**: DM pull-up detection status

This bit is active only during PD and gives the result of comparison between DM voltage level and VLGC 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, CDP or DCP)

1: PS2 port or proprietary charger detected

Bit 2 **SDET**: Secondary detection (SD) status

This bit gives the result of SD.

0: CDP detected

1: DCP detected

Bit 1 **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 CDP or DCP).

Bit 0 **DCDET**: Data contact detection (DCD) status

This bit gives the result of DCD.

0: data lines contact not detected

1: data lines contact detected

### 50.15.13 OTG core ID register (OTG\_CID)

Address offset: 0x03C

Reset value: 0x0000 3000

This is a register containing the Product ID as reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRODUCT_ID[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
PRODUCT_ID[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRODUCT\_ID[31:0]**: Product ID field  
Application-programmable ID field.

### 50.15.14 OTG core LPM configuration register (OTG\_GLPMCFG)

Address offset: 0x54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EN BESL	LPMRCNTSTS[2:0]			SND LPM	LPMRCNT[2:0]			LPMCHIDX[3:0]			L1RSM OK	
			rw	r	r	r	rs	rw	rw	rw	rw	rw	rw	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLP STS	LPMRSP[1:0]		L1DS EN	BESLTHRS[3:0]				L1SS EN	REM WAKE	BESL[3:0]				LPM ACK	LPM EN
r	r	r	rw	rw	rw	rw	rw	rw	rw/r	rw/r	rw/r	rw/r	rw/r	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ENBESL**: Enable best effort service latency

This bit enables the BESL feature as defined in the LPM errata:

0: The core works as described in the following document:

*USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007*

1: The core works as described in the LPM Errata:

*Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007*

*Note: Only the updated behavior (described in LPM Errata) is considered in this document and so the ENBESL bit should be set to '1' by application SW.*

Bits 27:25 **LPMRCNTSTS[2:0]**: LPM retry count status

Number of LPM host retries still remaining to be transmitted for the current LPM sequence.

*Note: Accessible only in host mode.*

Bit 24 **SNDLPM**: Send LPM transaction

When the application software sets this bit, an LPM transaction containing two tokens, EXT and LPM is sent. The hardware clears this bit once a valid response (STALL, NYET, or ACK) is received from the device or the core has finished transmitting the programmed number of LPM retries.

*Note: This bit must be set only when the host is connected to a local port.*

*Note: Accessible only in host mode.*

Bits 23:21 **LPMRCNT[2:0]** LPM retry count

When the device gives an ERROR response, this is the number of additional LPM retries that the host performs until a valid device response (STALL, NYET, or ACK) is received.

*Note: Accessible only in host mode.*

Bits 20:17 **LPMCHIDX[3:0]**: LPM Channel Index

The channel number on which the LPM transaction has to be applied while sending an LPM transaction to the local device. Based on the LPM channel index, the core automatically inserts the device address and endpoint number programmed in the corresponding channel into the LPM transaction.

*Note: Accessible only in host mode.*

Bit 16 **L1RSMOK**: Sleep state resume OK

Indicates that the device or host can start resume from Sleep state. This bit is valid in LPM sleep (L1) state. It is set in sleep mode after a delay of 50  $\mu$ s ( $T_{L1Residency}$ ).

This bit is reset when SLPSTS is 0.

1: The application or host can start resume from Sleep state

0: The application or host cannot start resume from Sleep state

Bit 15 **SLPSTS**: Port sleep status**Device mode:**

This bit is set as long as a Sleep condition is present on the USB bus. The core enters the Sleep state when an ACK response is sent to an LPM transaction and the  $T_{L1TokenRetry}$  timer has expired. To stop the PHY clock, the application must set the STPPCLK bit in OTG\_PCGCTL, which asserts the PHY suspend input signal.

The application must rely on SLPSTS and not ACK in LPMRSP to confirm transition into sleep.

The core comes out of sleep:

- When there is any activity on the USB linestate
- When the application writes to the RWUSIG bit in OTG\_DCTL or when the application resets or soft-disconnects the device.

**Host mode:**

The host transitions to Sleep (L1) state as a side-effect of a successful LPM transaction by the core to the local port with ACK response from the device. The read value of this bit reflects the current Sleep status of the port.

The core clears this bit after:

- The core detects a remote L1 wakeup signal,
- The application sets the PRST bit or the PRES bit in the OTG\_HPRT register, or
- The application sets the L1Resume/ remote wakeup detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT bit in OTG\_GINTSTS, respectively).

0: Core not in L1

1: Core in L1

Bits 14:13 **LPMRST[1:0]**: LPM response

**Device mode:**

The response of the core to LPM transaction received is reflected in these two bits.

**Host mode:**

Handshake response received from local device for LPM transaction

11: ACK

10: NYET

01: STALL

00: ERROR (No handshake response)

Bit 12 **L1DSEN**: L1 deep sleep enable

Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit should be set to '1' by application SW in all the cases.

Bits11:8 **BESLTHRS[3:0]**: BESL threshold

**Device mode:**

The core puts the PHY into deep low power mode in L1 when BESL value is greater than or equal to the value defined in this field BESL\_Thres[3:0].

**Host mode:**

The core puts the PHY into deep low power mode in L1. BESLTHRS[3:0] specifies the time for which resume signaling is to be reflected by host ( $T_{L1HubDrvResume2}$ ) on the USB bus when it detects device initiated resume.

BESLTHRS must not be programmed with a value greater than 1100b in host mode, because this exceeds maximum  $T_{L1HubDrvResume2}$ .

Thres[3:0] host mode resume signaling time ( $\mu$ s):

0000: 75

0001: 100

0010: 150

0011: 250

0100: 350

0101: 450

0110: 950

All other values: reserved

Bit 7 **L1SSEN**: L1 Shallow Sleep enable

Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit should be set to '1' by application SW in all the cases.

Bit 6 **REMWAKE**: bRemoteWake value

**Host mode:**

The value of remote wake up to be sent in the wIndex field of LPM transaction.

**Device mode (read-only):**

This field is updated with the received LPM token bRemoteWake bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.

Bits 5:2 **BESL[3:0]**: Best effort service latency

**Host mode:**

The value of BESL to be sent in an LPM transaction. This value is also used to initiate resume for a duration  $T_{L1HubDrvResume1}$  for host initiated resume.

**Device mode (read-only):**

This field is updated with the received LPM token BESL bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.

BESL[3:0] $T_{BESL}$  ( $\mu$ s)

- 0000: 125
- 0001: 150
- 0010: 200
- 0011: 300
- 0100: 400
- 0101: 500
- 0110: 1000
- 0111: 2000
- 1000: 3000
- 1001: 4000
- 1010: 5000
- 1011: 6000
- 1100: 7000
- 1101: 8000
- 1110: 9000
- 1111: 10000

Bit 1 **LPMACK**: LPM token acknowledge enable

Handshake response to LPM token preprogrammed by device application software.

1: ACK

Even though ACK is preprogrammed, the core device responds with ACK only on successful LPM transaction. The LPM transaction is successful if:

- No PID/CRC5 errors in either EXT token or LPM token (else ERROR)
- Valid bLinkState = 0001B (L1) received in LPM transaction (else STALL)
- No data pending in transmit queue (else NYET).

0: NYET

The preprogrammed software bit is over-ridden for response to LPM token when:

- The received bLinkState is not L1 (STALL response), or
- An error is detected in either of the LPM token packets because of corruption (ERROR response).

*Note: Accessible only in device mode.*

Bit 0 **LPMEN**: LPM support enable

The application uses this bit to control the OTG\_FS core LPM capabilities.

If the core operates as a non-LPM-capable host, it cannot request the connected device or hub to activate LPM mode.

If the core operates as a non-LPM-capable device, it cannot respond to any LPM transactions.

0: LPM capability is not enabled

1: LPM capability is enabled

### 50.15.15 OTG power down register (OTG\_GPWRDN)

Address offset: 0x058

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.	ADPIF	Res.	Res.	Res.	Res.	Res.	Res.	Res.
								rc_w1							
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.	ADPM EN
															rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ADPIF**: ADP interrupt flag

This bit is set whenever there is an ADP event

Bits 22:1 Reserved, must be kept at reset value.

Bit 0 **ADPMEN**: ADP module enable

This bit enables or disables the ADP logic.

0: Disable ADP module

1: Enable ADP module

### 50.15.16 OTG ADP timer, control and status register (OTG\_GADPCTL)

Address offset: 0x060

Reset value: 0x0000 0000

The OTG\_GADPCTL register must be accessed as follows:

- In order to read from the OTG\_GADPCTL register, program AR=0b01 and keep polling till AR=0b00. The core updates the other fields of this register and makes AR=0b00. Read values of this register are valid only when AR=0b00.
- In order to write to the OTG\_GADPCTL register, program AR=0b10 along with the values for the other fields and keep polling till AR=0b00. When AR becomes 0b00, it means that the programmed value has taken effect inside the core.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AR		ADP TOIM	ADP SNSIM	ADP PRBIM	ADP TOIF	ADP SNSIF	ADP PRBIF	ADPEN	ADP RST	ENA SNS	ENA PRB	RTIM
			rw	rw	rw	rw	rw	rc_w1	rc_w1	rc_w1	rw	rs	rw	rw	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTIM										PRBPER		PRBDELTA		PRBDSCHG	
r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw

- Bits 31:29 Reserved, must be kept at reset value.
- Bits 28:27 **AR**: Access request  
 This bitfield define the requested access mode to the OTG\_GADPCTL register:  
 00 Read / write valid (updated by the core)  
 01 Read request  
 10 Write request  
 11 Reserved
- Bit 26 **ADPTOIM**: ADP timeout interrupt mask  
 When this bit is set, it unmask the interrupt from ADPTOIF.
- Bit 25 **ADPSNSIM**: ADP sense interrupt mask  
 When this bit is set, it unmask the interrupt from ADPSNSIF.
- Bit 24 **ADPPRBIM**: ADP probe interrupt mask  
 When this bit is set, it unmask the interrupt from ADPPRBIF.
- Bit 23 **ADPTOIF**: ADP timeout interrupt flag  
 This bit is relevant only for an ADP probe. When this bit is set, it means that the ramp time has completed (RTIM has reached its terminal value of 0x7FF). This is a debug feature that allows the application to read the ramp time after each cycle.
- Bit 22 **ADPSNSIF**: ADP sense interrupt flag  
 When this bit is set, it means that the  $V_{BUS}$  voltage is greater than  $V_{ADPSNS}$  value or that  $V_{ADPSNS}$  is reached.
- Bit 21 **ADPPRBIF**: ADP probe interrupt flag  
 When this bit is set, it means that the  $V_{BUS}$  voltage is greater than  $V_{ADPPRB}$  or that  $V_{ADPPRB}$  is reached.
- Bit 20 **ADPEN**: ADP enable  
 When set, the core performs either ADP probing or sensing based on ENAPRB or ENASNS.
- Bit 19 **ADPRST**: ADP reset  
 When set, ADP controller is reset. This bit is auto-cleared after the reset procedure is complete in the ADP controller.
- Bit 18 **ENASNS**: Enable sense  
 When programmed to 1, the core performs a sense operation.
- Bit 17 **ENAPRB**: Enable probe  
 When programmed to 1, the core performs a probe operation.
- Bits 16:6 **RTIM**: Ramp time  
 These bits capture the latest time it took for  $V_{BUS}$  to ramp from  $V_{ADPSINK}$  to  $V_{ADPPRB}$ . The bits are defined in units of 32 kHz clock cycles as follow:  
 000: 1 cycle  
 001: 2 cycles  
 002: 3 cycles  
 ...  
 7FF: 2048 cycles  
 A time of 1024 cycles at 32 kHz corresponds to a time of 32 ms.



Bits 5:4 **PRBPER**: Probe period

- These bits sets the  $T_{ADPPRD}$  as follow:
- 00: 0.625 to 0.925 sec (typical 0.775 sec)
- 01: 1.25 to 1.85 sec (typical 1.55 sec)
- 10: 1.9 to 2.6 sec (typical 2.275 sec)
- 11: Reserved

Bits 3:2 **PRBDELTA**: Probe delta

- These bits set the resolution for RTIM value. The bits are defined in units of 32 kHz clock cycles as follow:
- 00: 1 cycle
- 01: 2 cycles
- 10: 3 cycles
- 11: 4 cycles
- For example if this value is chosen to be 01, it means that RTIM increments for every two 32 kHz clock cycles.

Bits 1:0 **PRBDSCHG**: Probe discharge

- These bits set the times for  $T_{ADP\_DSCHG}$ . These bits are defined as follow:
- 00: 4 ms
- 01: 8 ms
- 10: 16 ms
- 11: 32 ms

### 50.15.17 OTG host periodic transmit FIFO size register (OTG\_HPTXFSIZ)

Address offset: 0x100

Reset value: 0x0200 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTXFSIZ[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXSA[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:16 **PTXFSIZ[15:0]**: Host periodic Tx FIFO depth

- This value is in terms of 32-bit words.
- Minimum value is 16

Bits 15:0 **PTXSA[15:0]**: Host periodic Tx FIFO start address

- This field configures the memory start address for periodic transmit FIFO RAM.

### 50.15.18 OTG device IN endpoint transmit FIFO size register (OTG\_DIEPTXF<sub>x</sub>) (x = 1..5, where x is the FIFO number)

Address offset: 0x104 + (x – 1) \* 0x04

Reset value: 0x0200 0200 + (x \* 0x200)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INEPTXFD[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
INEPTXSA[15:0]															
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Bits 31:16 **INEPTXFD[15:0]**: IN endpoint Tx FIFO depth

This value is in terms of 32-bit words.

Minimum value is 16

Bits 15:0 **INEPTXSA[15:0]**: IN endpoint FIFOx transmit RAM start address

This field contains the memory start address for IN endpoint transmit FIFOx. The address must be aligned with a 32-bit memory location.

### 50.15.19 Host-mode registers

Bit values in the register descriptions are expressed in binary unless otherwise specified.

Host-mode registers affect the operation of the core in the host mode. Host mode registers must not be accessed in device mode, as the results are undefined. Host mode registers can be categorized as follows:

### 50.15.20 OTG host configuration register (OTG\_HCFG)

Address offset: 0x400

Reset value: 0x0000 0000

This register configures the core after power-on. Do not make changes to this register after initializing the host.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	FSLSS	FSLSPCS[1:0]	
													r	rW	rW

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FSLSS**: FS- and LS-only support

The application uses this bit to control the core’s enumeration speed. Using this bit, the application can make the core enumerate as an FS host, even if the connected device supports HS traffic. Do not make changes to this field after initial programming.

1: FS/LS-only, even if the connected device can support HS (read-only).

Bits 1:0 **FSLSPCS[1:0]**: FS/LS PHY clock select

When the core is in FS host mode

01: PHY clock is running at 48 MHz

Others: Reserved

When the core is in LS host mode

00: Reserved

01: Select 48 MHz PHY clock frequency

10: Select 6 MHz PHY clock frequency

11: Reserved

*Note: The FSLSPCS must be set on a connection event according to the speed of the connected device (after changing this bit, a software reset must be performed).*

### 50.15.21 OTG host frame interval register (OTG\_HFIR)

Address offset: 0x404

Reset value: 0x0000 EA60

This register stores the frame interval information for the current speed to which the OTG\_FS controller has enumerated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLD CTRL
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRIVL[15:0]															
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.

Bit 16 **RLDCTRL**: Reload control

This bit allows dynamic reloading of the HFIR register during run time.

0: The HFIR can be dynamically reloaded during run time.

1: The HFIR cannot be reloaded dynamically

This bit needs to be programmed during initial configuration and its value must not be changed during run time.

**Caution:** RLDCTRL = 1 is not recommended.

Bits 15:0 **FRIVL[15:0]**: Frame interval

The value that the application programs to this field, specifies the interval between two consecutive SOFs (FS) or Keep-Alive tokens (LS). This field contains the number of PHY clocks that constitute the required frame interval. The application can write a value to this register only after the port enable bit of the host port control and status register (PENA bit in OTG\_HPRT) has been set. If no value is programmed, the core calculates the value based on the PHY clock specified in the FS/LS PHY clock select field of the host configuration register (FSLSPCS in OTG\_HCFG). Do not change the value of this field after the initial configuration, unless the RLDCTRL bit is set. In such case, the FRIVL is reloaded with each SOF event.

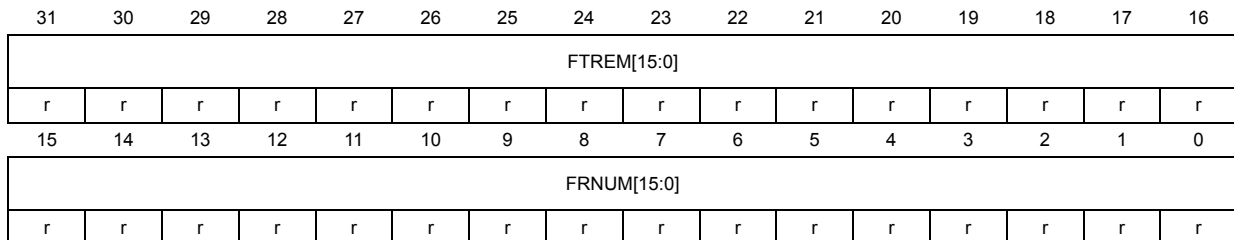
– Frame interval = 1 ms × (FRIVL - 1)

### 50.15.22 OTG host frame number/frame time remaining register (OTG\_HFNUM)

Address offset: 0x408

Reset value: 0x0000 3FFF

This register indicates the current frame number. It also indicates the time remaining (in terms of the number of PHY clocks) in the current frame.



Bits 31:16 **FTREM[15:0]**: Frame time remaining

Indicates the amount of time remaining in the current frame, in terms of PHY clocks. This field decrements on each PHY clock. When it reaches zero, this field is reloaded with the value in the Frame interval register and a new SOF is transmitted on the USB.

Bits 15:0 **FRNUM[15:0]**: Frame number

This field increments when a new SOF is transmitted on the USB, and is cleared to 0 when it reaches 0x3FFF.

### 50.15.23 OTG\_Host periodic transmit FIFO/queue status register (OTG\_HPTXSTS)

Address offset: 0x410

Reset value: 0x0008 0100

This read-only register contains the free space information for the periodic Tx FIFO and the periodic transmit request queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTXQTOP[7:0]								PTXQSAV[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXFSAVL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **PTXQTOP[7:0]**: Top of the periodic transmit request queue

This indicates the entry in the periodic Tx request queue that is currently being processed by the MAC.

This register is used for debugging.

Bit 31: Odd/Even frame

0: send in even frame

1: send in odd frame

Bits 30:27: Channel/endpoint number

Bits 26:25: Type

00: IN/OUT

01: Zero-length packet

11: Disable channel command

Bit 24: Terminate (last entry for the selected channel/endpoint)

Bits 23:16 **PTXQSAV[7:0]**: Periodic transmit request queue space available

Indicates the number of free locations available to be written in the periodic transmit request queue. This queue holds both IN and OUT requests.

00: Periodic transmit request queue is full

01: 1 location available

10: 2 locations available

bxn: n locations available ( $0 \leq n \leq 8$ )

Others: Reserved

Bits 15:0 **PTXFSAVL[15:0]**: Periodic transmit data FIFO space available

Indicates the number of free locations available to be written to in the periodic Tx FIFO.

Values are in terms of 32-bit words

0000: Periodic Tx FIFO is full

0001: 1 word available

0010: 2 words available

bxn: n words available (where  $0 \leq n \leq \text{PTXFD}$ )

Others: Reserved

### 50.15.24 OTG host all channels interrupt register (OTG\_HAINT)

Address offset: 0x414

Reset value: 0x0000 0000

When a significant event occurs on a channel, the host all channels interrupt register interrupts the application using the host channels interrupt bit of the core interrupt register (HCINT bit in OTG\_GINTSTS). This is shown in *Figure 588*. There is one interrupt bit per channel, up to a maximum of 16 bits. Bits in this register are set and cleared when the application sets and clears bits in the corresponding host channel-x interrupt register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HAINT[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 **HAINT[15:0]**: Channel interrupts

One bit per channel: Bit 0 for Channel 0, bit 15 for Channel 15

### 50.15.25 OTG host all channels interrupt mask register (OTG\_HAINTMSK)

Address offset: 0x418

Reset value: 0x0000 0000

The host all channel interrupt mask register works with the host all channel interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per channel, up to a maximum of 16 bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HAINTM[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **HAINTM[15:0]**: Channel interrupt mask

0: Masked interrupt

1: Unmasked interrupt

One bit per channel: Bit 0 for channel 0, bit 15 for channel 15

### 50.15.26 OTG host port control and status register (OTG\_HPRT)

Address offset: 0x440

Reset value: 0x0000 0000

This register is available only in host mode. Currently, the OTG host supports only one port.

A single register holds USB port-related information such as USB reset, enable, suspend, resume, connect status, and test mode for each port. It is shown in *Figure 588*. The rc\_w1 bits in this register can trigger an interrupt to the application through the host port interrupt bit of the core interrupt register (HPRTINT bit in OTG\_GINTSTS). On a port interrupt, the application must read this register and clear the bit that caused the interrupt. For the rc\_w1 bits, the application must write a 1 to the bit to clear the interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSPD[1:0]		PTCTL [3]
													r	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTCTL[2:0]			PPWR	PLSTS[1:0]		Res.	PRST	PSUSP	PRES	POC CHNG	POCA	PEN CHNG	PENA	PCDET	PCSTS
rw	rw	rw	rw	r	r		rw	rs	rw	rc_w1	r	rc_w1	rc_w1	rc_w1	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 **PSPD[1:0]**: Port speed

Indicates the speed of the device attached to this port.

- 01: Full speed
- 10: Low speed
- 11: Reserved

Bits 16:13 **PTCTL[3:0]**: Port test control

The application writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is signaled on the port.

- 0000: Test mode disabled
- 0001: Test\_J mode
- 0010: Test\_K mode
- 0011: Test\_SE0\_NAK mode
- 0100: Test\_Packet mode
- 0101: Test\_Force\_Enable
- Others: Reserved

Bit 12 **PPWR**: Port power

The application uses this field to control power to this port, and the core clears this bit on an overcurrent condition.

- 0: Power off
- 1: Power on

Bits 11:10 **PLSTS[1:0]**: Port line status

Indicates the current logic level USB data lines

- Bit 10: Logic level of OTG\_DP
- Bit 11: Logic level of OTG\_DM

Bit 9 Reserved, must be kept at reset value.



**Bit 8 PRST:** Port reset

When the application sets this bit, a reset sequence is started on this port. The application must time the reset period and clear this bit after the reset sequence is complete.

0: Port not in reset

1: Port in reset

The application must leave this bit set for a minimum duration of at least 10 ms to start a reset on the port. The application can leave it set for another 10 ms in addition to the required minimum duration, before clearing the bit, even though there is no maximum limit set by the USB standard.

High speed: 50 ms

Full speed/Low speed: 10 ms

**Bit 7 PSUSP:** Port suspend

The application sets this bit to put this port in suspend mode. The core only stops sending SOFs when this is set. To stop the PHY clock, the application must set the port clock stop bit, which asserts the suspend input pin of the PHY.

The read value of this bit reflects the current suspend status of the port. This bit is cleared by the core after a remote wakeup signal is detected or the application sets the port reset bit or port resume bit in this register or the resume/remote wakeup detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT in OTG\_GINTSTS, respectively).

0: Port not in suspend mode

1: Port in suspend mode

**Bit 6 PRES:** Port resume

The application sets this bit to drive resume signaling on the port. The core continues to drive the resume signal until the application clears this bit.

If the core detects a USB remote wakeup sequence, as indicated by the port resume/remote wakeup detected interrupt bit of the core interrupt register (WKUPINT bit in OTG\_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit when it detects a disconnect condition. The read value of this bit indicates whether the core is currently driving resume signaling.

0: No resume driven

1: Resume driven

When LPM is enabled and the core is in L1 state, the behavior of this bit is as follow:

1. The application sets this bit to drive resume signaling on the port.

2. The core continues to drive the resume signal until a predetermined time specified in BESLTHRS[3:0] field of OTG\_GLPMCFG register.

3. If the core detects a USB remote wakeup sequence, as indicated by the port L1Resume/Remote L1Wakeup detected interrupt bit of the core interrupt register (WKUPINT in OTG\_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit at the end of resume. This bit can be set or cleared by both the core and the application. This bit is cleared by the core even if there is no device connected to the host.

**Bit 5 POCCHNG:** Port overcurrent change

The core sets this bit when the status of the port overcurrent active bit (bit 4) in this register changes.

**Bit 4 POCA:** Port overcurrent active

Indicates the overcurrent condition of the port.

0: No overcurrent condition

1: Overcurrent condition

**Bit 3 PENCHNG:** Port enable/disable change

The core sets this bit when the status of the port enable bit 2 in this register changes.



Bit 2 **PENA**: Port enable

A port is enabled only by the core after a reset sequence, and is disabled by an overcurrent condition, a disconnect condition, or by the application clearing this bit. The application cannot set this bit by a register write. It can only clear it to disable the port. This bit does not trigger any interrupt to the application.

0: Port disabled  
1: Port enabled

Bit 1 **PCDET**: Port connect detected

The core sets this bit when a device connection is detected to trigger an interrupt to the application using the host port interrupt bit in the core interrupt register (HPRTINT bit in OTG\_GINTSTS). The application must write a 1 to this bit to clear the interrupt.

Bit 0 **PCSTS**: Port connect status

0: No device is attached to the port  
1: A device is attached to the port

### 50.15.27 OTG host channel x characteristics register (OTG\_HCCHARx) (x = 0..11, where x = Channel number)

Address offset: 0x500 + (x \* 0x20)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CHENA	CHDIS	ODD FRM	DAD[6:0]						MCNT[1:0]		EPTYP[1:0]		LSDEV	Res.	
rs	rs	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
EPDIR	EPNUM[3:0]			MPSIZ[10:0]											
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CHENA**: Channel enable

This field is set by the application and cleared by the OTG host.

0: Channel disabled  
1: Channel enabled

Bit 30 **CHDIS**: Channel disable

The application sets this bit to stop transmitting/receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel disabled interrupt before treating the channel as disabled.

Bit 29 **ODDFRM**: Odd frame

This field is set (reset) by the application to indicate that the OTG host must perform a transfer in an odd frame. This field is applicable for only periodic (isochronous and interrupt) transactions.

0: Even frame  
1: Odd frame

Bits 28:22 **DAD[6:0]**: Device address

This field selects the specific device serving as the data source or sink.

Bits 21:20 **MCNT[1:0]**: Multicount

This field indicates to the host the number of transactions that must be executed per frame for this periodic endpoint. For non-periodic transfers, this field is not used

- 00: Reserved. This field yields undefined results
- 01: 1 transaction
- 10: 2 transactions per frame to be issued for this endpoint
- 11: 3 transactions per frame to be issued for this endpoint

*Note: This field must be set to at least 01.*

Bits 19:18 **EPTYP[1:0]**: Endpoint type

Indicates the transfer type selected.

- 00: Control
- 01: Isochronous
- 10: Bulk
- 11: Interrupt

Bit 17 **LSDEV**: Low-speed device

This field is set by the application to indicate that this channel is communicating to a low-speed device.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **EPDIR**: Endpoint direction

Indicates whether the transaction is IN or OUT.

- 0: OUT
- 1: IN

Bits 14:11 **EPNUM[3:0]**: Endpoint number

Indicates the endpoint number on the device serving as the data source or sink.

Bits 10:0 **MPSIZ[10:0]**: Maximum packet size

Indicates the maximum packet size of the associated endpoint.

**50.15.28 OTG host channel x interrupt register (OTG\_HCINTx)  
(x = 0..11, where x = Channel number)**

Address offset: 0x508 + (x \* 0x20)

Reset value: 0x0000 0000

This register indicates the status of a channel with respect to USB- and AHB-related events. It is shown in [Figure 588](#). The application must read this register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG\_GINTSTS) is set. Before the application can read this register, it must first read the host all channels interrupt (OTG\_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG\_HAINT and OTG\_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	DTERR	FRM OR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
					rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DTERR**: Data toggle error.

Bit 9 **FRMOR**: Frame overrun.

Bit 8 **BBERR**: Babble error.

Bit 7 **TXERR**: Transaction error.

- Indicates one of the following errors occurred on the USB.
- CRC check failure
- Timeout
- Bit stuff error
- False EOP

Bit 6 Reserved, must be kept at reset value.

Bit 5 **ACK**: ACK response received/transmitted interrupt.

Bit 4 **NAK**: NAK response received interrupt.

Bit 3 **STALL**: STALL response received interrupt.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CHH**: Channel halted.

- Indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application.

Bit 0 **XFRC**: Transfer completed.

- Transfer completed normally without any errors.

### 50.15.29 OTG host channel x interrupt mask register (OTG\_HCINTMSKx) (x = 0..11, where x = Channel number)

Address offset: 0x50C + (x \* 0x20)

Reset value: 0x0000 0000

This register reflects the mask for each channel status described in the previous section.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	DTERR M	FRM ORM	BBERR M	TXERR M	Res.	ACKM	NAKM	STALL M	Res.	CHHM	XFRC M
					rw	rw	rw	rw		rw	rw	rw		rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DTERRM**: Data toggle error mask.

- 0: Masked interrupt
- 1: Unmasked interrupt

Bit 9 **FRMORM**: Frame overrun mask.

- 0: Masked interrupt
- 1: Unmasked interrupt

Bit 8 **BBERRM**: Babble error mask.

- 0: Masked interrupt
- 1: Unmasked interrupt

Bit 7 **TXERRM**: Transaction error mask.

- 0: Masked interrupt
- 1: Unmasked interrupt

Bit 6 Reserved, must be kept at reset value.

Bit 5 **ACKM**: ACK response received/transmitted interrupt mask.

- 0: Masked interrupt
- 1: Unmasked interrupt

Bit 4 **NAKM**: NAK response received interrupt mask.

- 0: Masked interrupt
- 1: Unmasked interrupt

Bit 3 **STALLM**: STALL response received interrupt mask.

- 0: Masked interrupt
- 1: Unmasked interrupt

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CHHM**: Channel halted mask

- 0: Masked interrupt
- 1: Unmasked interrupt

Bit 0 **XFRM**: Transfer completed mask

- 0: Masked interrupt
- 1: Unmasked interrupt

### 50.15.30 OTG host channel x transfer size register (OTG\_HCTSIZx) (x = 0..11, where x = Channel number)

Address offset: 0x510 + (x \* 0x20)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DPID[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
	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
XFRSIZ[15:0]															
rw	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 30:29 **DPID[1:0]**: Data PID

The application programs this field with the type of PID to use for the initial transaction. The host maintains this field for the rest of the transfer.

00: DATA0

10: DATA1

11: SETUP (control) / reserved (non-control)

Bits 28:19 **PKTCNT[9:0]**: Packet count

This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN).

The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion.

Bits 18:0 **XFRSIZ[18:0]**: Transfer size

For an OUT, this field is the number of data bytes the host sends during the transfer.

For an IN, this field is the buffer size that the application has reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).

### 50.15.31 Device-mode registers

These registers must be programmed every time the core changes to device mode

### 50.15.32 OTG device configuration register (OTG\_DCFG)

Address offset: 0x800

Reset value: 0x0220 0000

This register configures the core in device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRATIM	Res.	Res.	PFIVL[1:0]		DAD[6:0]						Res.	NZLSO HSK	DSPD[1:0]		
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 **ERRATIM**: Erratic error interrupt mask

1: Mask early suspend interrupt on erratic error

0: Early suspend interrupt is generated on erratic error

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.



Bits 12:11 **PFIVL[1:0]**: Periodic frame interval

Indicates the time within a frame at which the application must be notified using the end of periodic frame interrupt. This can be used to determine if all the isochronous traffic for that frame is complete.

- 00: 80% of the frame interval
- 01: 85% of the frame interval
- 10: 90% of the frame interval
- 11: 95% of the frame interval

Bits 10:4 **DAD[6:0]**: Device address

The application must program this field after every SetAddress control command.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **NZLSOHSK**: Non-zero-length status OUT handshake

The application can use this field to select the handshake the core sends on receiving a non-zero-length data packet during the OUT transaction of a control transfer's status stage.  
 1: Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application.

0: Send the received OUT packet to the application (zero-length or nonzero-length) and send a handshake based on the NAK and STALL bits for the endpoint in the device endpoint control register.

Bits 1:0 **DSPD[1:0]**: Device speed

Indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected.

- 00: Reserved
- 01: Reserved
- 10: Reserved
- 11: Full speed (USB 1.1 transceiver clock is 48 MHz)

### 50.15.33 OTG device control register (OTG\_DCTL)

Address offset: 0x804

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DS BESL RJCT	Res.	Res.
													rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PO PRG DNE	CGO NAK	SGO NAK	CGI NAK	SGI NAK	TCTL[2:0]			GON STS	GIN STS	SDIS	RWU SIG
				rw	w	w	w	w	rw	rw	rw	r	r	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **DSBESLRJCT**: Deep sleep BESL reject

Core rejects LPM request with BESL value greater than BESL threshold programmed. NYET response is sent for LPM tokens with BESL value greater than BESL threshold. By default, the deep sleep BESL reject feature is disabled.

Bits 17:12 Reserved, must be kept at reset value.

Bit 11 **POPRGDNE**: Power-on programming done

The application uses this bit to indicate that register programming is completed after a wakeup from power down mode.

Bit 10 **CGONAK**: Clear global OUT NAK

Writing 1 to this field clears the Global OUT NAK.

Bit 9 **SGONAK**: Set global OUT NAK

Writing 1 to this field sets the Global OUT NAK. The application uses this bit to send a NAK handshake on all OUT endpoints. The application must set this bit only after making sure that the Global OUT NAK effective bit in the core interrupt register (GONAKEFF bit in OTG\_GINTSTS) is cleared.

Bit 8 **CGINAK**: Clear global IN NAK

Writing 1 to this field clears the Global IN NAK.

Bit 7 **SGINAK**: Set global IN NAK

Writing 1 to this field sets the Global non-periodic IN NAK. The application uses this bit to send a NAK handshake on all non-periodic IN endpoints. The application must set this bit only after making sure that the Global IN NAK effective bit in the core interrupt register (GINAKEFF bit in OTG\_GINTSTS) is cleared.

Bits 6:4 **TCTL[2:0]**: Test control

000: Test mode disabled  
001: Test\_J mode  
010: Test\_K mode  
011: Test\_SE0\_NAK mode  
100: Test\_Packet mode  
101: Test\_Force\_Enable  
Others: Reserved

Bit 3 **GONSTS**: Global OUT NAK status

0: A handshake is sent based on the FIFO status and the NAK and STALL bit settings.  
1: No data is written to the Rx FIFO, irrespective of space availability. Sends a NAK handshake on all packets, except on SETUP transactions. All isochronous OUT packets are dropped.

Bit 2 **GINSTS**: Global IN NAK status

0:A handshake is sent out based on the data availability in the transmit FIFO.

1:A NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO.

Bit 1 **SDIS**: Soft disconnect

The application uses this bit to signal the USB OTG core to perform a soft disconnect. As long as this bit is set, the host does not see that the device is connected, and the device does not receive signals on the USB. The core stays in the disconnected state until the application clears this bit.

0:Normal operation. When this bit is cleared after a soft disconnect, the core generates a device connect event to the USB host. When the device is reconnected, the USB host restarts device enumeration.

1:The core generates a device disconnect event to the USB host.

Bit 0 **RWUSIG**: Remote wakeup signaling

When the application sets this bit, the core initiates remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the suspend state. As specified in the USB 2.0 specification, the application must clear this bit 1 ms to 15 ms after setting it.

If LPM is enabled and the core is in the L1 (sleep) state, when the application sets this bit, the core initiates L1 remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the sleep state. As specified in the LPM specification, the hardware automatically clears this bit 50  $\mu$ s ( $T_{L1DevDrvResume}$ ) after being set by the application. The application must not set this bit when bRemoteWake from the previous LPM transaction is zero (refer to REMWAKE bit in GLPMCFG register).

[Table 328](#) contains the minimum duration (according to device state) for which the Soft disconnect (SDIS) bit must be set for the USB host to detect a device disconnect. To accommodate clock jitter, it is recommended that the application add some extra delay to the specified minimum duration.

**Table 328. Minimum duration for soft disconnect**

Operating speed	Device state	Minimum duration
Full speed	Suspended	1 ms + 2.5 $\mu$ s
Full speed	Idle	2.5 $\mu$ s
Full speed	Not Idle or suspended (Performing transactions)	2.5 $\mu$ s



### 50.15.34 OTG device status register (OTG\_DSTS)

Address offset: 0x808

Reset value: 0x0000 0010

This register indicates the status of the core with respect to USB-related events. It must be read on interrupts from the device all interrupts (OTG\_DAINTE) register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVLNSTS[1:0]		FNSOF[13:8]					
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FNSOF[7:0]								Res.	Res.	Res.	Res.	EERR	ENUMSPD[1:0]		SUSPSTS
r	r	r	r	r	r	r	r					r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **DEVLNSTS[1:0]**: Device line status  
 Indicates the current logic level USB data lines.  
 Bit [23]: Logic level of D+  
 Bit [22]: Logic level of D-

Bits 21:8 **FNSOF[13:0]**: Frame number of the received SOF

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **EERR**: Erratic error  
 The core sets this bit to report any erratic errors.  
 Due to erratic errors, the OTG\_FS controller goes into suspended state and an interrupt is generated to the application with Early suspend bit of the OTG\_GINTSTS register (ESUSP bit in OTG\_GINTSTS). If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover.

Bits 2:1 **ENUMSPD[1:0]**: Enumerated speed  
 Indicates the speed at which the OTG\_FS controller has come up after speed detection through a chirp sequence.  
 01: Reserved  
 10: Reserved  
 11: Full speed (PHY clock is running at 48 MHz)  
 Others: reserved

Bit 0 **SUSPSTS**: Suspend status  
 In device mode, this bit is set as long as a suspend condition is detected on the USB. The core enters the suspended state when there is no activity on the USB data lines for a period of 3 ms. The core comes out of the suspend:  
 – When there is an activity on the USB data lines  
 – When the application writes to the remote wakeup signaling bit in the OTG\_DCTL register (RWUSIG bit in OTG\_DCTL).

### 50.15.35 OTG device IN endpoint common interrupt mask register (OTG\_DIEPMSK)

Address offset: 0x810

Reset value: 0x0000 0000

This register works with each of the OTG\_DIEPINTx registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the OTG\_DIEPINTx register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAKM	Res.	Res.	Res.	Res.	TXFURM	Res.	INEPNEM	INEPNMM	ITTXFEMSK	TOM	Res.	EPDM	XFRCM
		rw					rw		rw	rw	rw	rw		rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAKM**: NAK interrupt mask

0: Masked interrupt

1: Unmasked interrupt

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **TXFURM**: FIFO underrun mask

0: Masked interrupt

1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **INEPNEM**: IN endpoint NAK effective mask

0: Masked interrupt

1: Unmasked interrupt

Bit 5 **INEPNMM**: IN token received with EP mismatch mask

0: Masked interrupt

1: Unmasked interrupt

Bit 4 **ITTXFEMSK**: IN token received when Tx FIFO empty mask

0: Masked interrupt

1: Unmasked interrupt

Bit 3 **TOM**: Timeout condition mask (Non-isochronous endpoints)

0: Masked interrupt

1: Unmasked interrupt

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EPDM**: Endpoint disabled interrupt mask  
 0: Masked interrupt  
 1: Unmasked interrupt

Bit 0 **XFRCM**: Transfer completed interrupt mask  
 0: Masked interrupt  
 1: Unmasked interrupt

### 50.15.36 OTG device OUT endpoint common interrupt mask register (OTG\_DOEPMSK)

Address offset: 0x814

Reset value: 0x0000 0000

This register works with each of the OTG\_DOEPINTx registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the OTG\_DOEPINTx register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NYET MSK	NAK MSK	BERR M	Res.	Res.	Res.	OUT PKT ERRM	Res.	Res.	STS PHSR XM	OTEPD M	STUPM	Res.	EPDM	XFRC M
	rw	rw	rw				rw			rw	rw	rw		rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **NYETMSK**: NYET interrupt mask  
 0: Masked interrupt  
 1: Unmasked interrupt

Bit 13 **NAKMSK**: NAK interrupt mask  
 0: Masked interrupt  
 1: Unmasked interrupt

Bit 12 **BERRM**: Babble error interrupt mask  
 0: Masked interrupt  
 1: Unmasked interrupt

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **OUTPKTERRM**: Out packet error mask  
 0: Masked interrupt  
 1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.



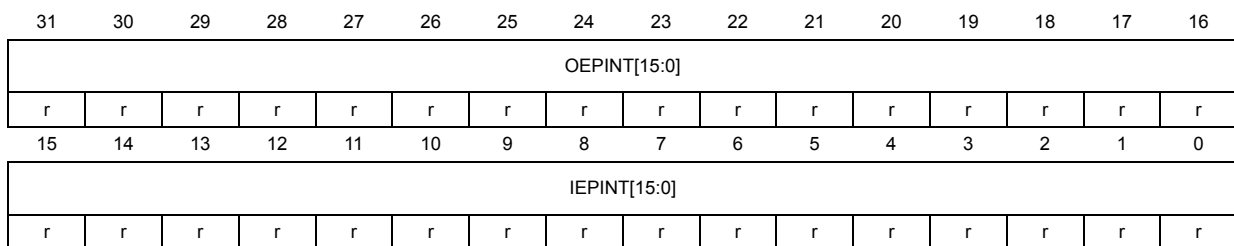
- Bit 5 **STSPHSRXM**: Status phase received for control write mask
  - 0: Masked interrupt
  - 1: Unmasked interrupt
- Bit 4 **OTEPDM**: OUT token received when endpoint disabled mask. Applies to control OUT endpoints only.
  - 0: Masked interrupt
  - 1: Unmasked interrupt
- Bit 3 **STUPM**: STUPM: SETUP phase done mask. Applies to control endpoints only.
  - 0: Masked interrupt
  - 1: Unmasked interrupt
- Bit 2 Reserved, must be kept at reset value.
- Bit 1 **EPDM**: Endpoint disabled interrupt mask
  - 0: Masked interrupt
  - 1: Unmasked interrupt
- Bit 0 **XFRM**: Transfer completed interrupt mask
  - 0: Masked interrupt
  - 1: Unmasked interrupt

### 50.15.37 OTG device all endpoints interrupt register (OTG\_DAIN)

Address offset: 0x818

Reset value: 0x0000 0000

When a significant event occurs on an endpoint, a OTG\_DAIN register interrupts the application using the device OUT endpoints interrupt bit or device IN endpoints interrupt bit of the OTG\_GINTSTS register (OEPINT or IEPINT in OTG\_GINTSTS, respectively). There is one interrupt bit per endpoint, up to a maximum of 16 bits for OUT endpoints and 16 bits for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used. Bits in this register are set and cleared when the application sets and clears bits in the corresponding device endpoint-x interrupt register (OTG\_DIEPINTx/OTG\_DOEPINTx).



- Bits 31:16 **OEPINT[15:0]**: OUT endpoint interrupt bits
  - One bit per OUT endpoint:
  - Bit 16 for OUT endpoint 0, bit 19 for OUT endpoint 3.
- Bits 15:0 **IEPINT[15:0]**: IN endpoint interrupt bits
  - One bit per IN endpoint:
  - Bit 0 for IN endpoint 0, bit 3 for endpoint 3.



### 50.15.38 OTG all endpoints interrupt mask register (OTG\_DAINMSK)

Address offset: 0x81C

Reset value: 0x0000 0000

The OTG\_DAINMSK register works with the device endpoint interrupt register to interrupt the application when an event occurs on a device endpoint. However, the OTG\_DAIN register bit corresponding to that interrupt is still set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEPM[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
IEPM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **OEPM[15:0]**: OUT EP interrupt mask bits  
 One per OUT endpoint:  
 Bit 16 for OUT EP 0, bit 19 for OUT EP 3  
 0: Masked interrupt  
 1: Unmasked interrupt

Bits 15:0 **IEPM[15:0]**: IN EP interrupt mask bits  
 One bit per IN endpoint:  
 Bit 0 for IN EP 0, bit 3 for IN EP 3  
 0: Masked interrupt  
 1: Unmasked interrupt

### 50.15.39 OTG device V<sub>BUS</sub> discharge time register (OTG\_DVBUSDIS)

Address offset: 0x0828

Reset value: 0x0000 17D7

This register specifies the V<sub>BUS</sub> discharge time after V<sub>BUS</sub> pulsing during SRP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VBUSDT[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 **VBUSDT[15:0]**: Device  $V_{BUS}$  discharge time  
 Specifies the  $V_{BUS}$  discharge time after  $V_{BUS}$  pulsing during SRP. This value equals:  
 $V_{BUS}$  discharge time in PHY clocks / 1 024  
 Depending on your  $V_{BUS}$  load, this value may need adjusting.

**50.15.40 OTG device  $V_{BUS}$  pulsing time register (OTG\_DVBUSPULSE)**

Address offset: 0x082C

Reset value: 0x0000 05B8

This register specifies the  $V_{BUS}$  pulsing time during SRP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DVBUSP[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DVBUSP[15:0]**: Device  $V_{BUS}$  pulsing time. This feature is only relevant to OTG1.3.  
 Specifies the  $V_{BUS}$  pulsing time during SRP. This value equals:  
 $V_{BUS}$  pulsing time in PHY clocks / 1 024

**50.15.41 OTG device IN endpoint FIFO empty interrupt mask register (OTG\_DIEPEMPMSK)**

Address offset: 0x834

Reset value: 0x0000 0000

This register is used to control the IN endpoint FIFO empty interrupt generation (TXFE\_OTG\_DIEPINTx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTXFEM[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w



Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INEPTXFEM[15:0]**: IN EP Tx FIFO empty interrupt mask bits

These bits act as mask bits for OTG\_DIEPINTx.

TXFE interrupt one bit per IN endpoint:

Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3

0: Masked interrupt

1: Unmasked interrupt

### 50.15.42 OTG device control IN endpoint 0 control register (OTG\_DIEPCTL0)

Address offset: 0x900

Reset value: 0x0000 0000

This section describes the OTG\_DIEPCTL0 register for USB\_OTG FS. Nonzero control endpoints use registers for endpoints 1–3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	Res.	Res.	SNAK	CNAK	TXFNUM[3:0]				STALL	Res.	EPTYP		NAK STS	Res.
rs	rs			w	w	rw	rw	rw	rw	rs		r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ[1:0]	
r														rw	rw

Bit 31 **EPENA**: Endpoint enable

The application sets this bit to start transmitting data on the endpoint 0.

The core clears this bit before setting any of the following interrupts on this endpoint:

- Endpoint disabled
- Transfer completed

Bit 30 **EPDIS**: Endpoint disable

The application sets this bit to stop transmitting data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **SNAK**: Set NAK

A write to this bit sets the NAK bit for the endpoint.

Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for an endpoint after a SETUP packet is received on that endpoint.

Bit 26 **CNAK**: Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 **TXFNUM[3:0]**: Tx FIFO number

This value is set to the FIFO number that is assigned to IN endpoint 0.

Bit 21 **STALL**: STALL handshake

The application can only set this bit, and the core clears it when a SETUP token is received for this endpoint. If a NAK bit, a Global IN NAK or Global OUT NAK is set along with this bit, the STALL bit takes priority.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP**: Endpoint type

Hardcoded to '00' for control.

Bit 17 **NAKSTS**: NAK status

Indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status

1: The core is transmitting NAK handshakes on this endpoint.

When this bit is set, either by the application or core, the core stops transmitting data, even if there are data available in the Tx FIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **USBAEP**: USB active endpoint

This bit is always set to 1, indicating that control endpoint 0 is always active in all configurations and interfaces.

Bits 14:2 Reserved, must be kept at reset value.

Bits 1:0 **MPSIZ[1:0]**: Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint.

00: 64 bytes

01: 32 bytes

10: 16 bytes

11: 8 bytes

### 50.15.43 OTG device IN endpoint x control register (OTG\_DIEPCTLx) (x = 1..5 , where x = endpoint number)

Address offset: 0x900 + (x \* 0x20)

Reset value: 0x0000 0000

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SODD FRM	SD0 PID/ SEVN FRM	SNAK	CNAK	TXFNUM[3:0]				STALL	Res.	EPTYP[1:0]		NAK STS	EO NUM/ DPID
rs	rs	w	w	w	w	rw	rw	rw	rw	rw/rs		rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw



- Bit 31 **EPENA**: Endpoint enable  
The application sets this bit to start transmitting data on an endpoint.  
The core clears this bit before setting any of the following interrupts on this endpoint:
- SETUP phase done
  - Endpoint disabled
  - Transfer completed
- Bit 30 **EPDIS**: Endpoint disable  
The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.
- Bit 29 **SODDFRM**: Set odd frame  
Applies to isochronous IN and OUT endpoints only.  
Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.
- Bit 28 **SD0PID**: Set DATA0 PID  
Applies to interrupt/bulk IN endpoints only.  
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.
- SEVNFRM**: Set even frame  
Applies to isochronous IN endpoints only.  
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.
- Bit 27 **SNAK**: Set NAK  
A write to this bit sets the NAK bit for the endpoint.  
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.
- Bit 26 **CNAK**: Clear NAK  
A write to this bit clears the NAK bit for the endpoint.
- Bits 25:22 **TXFNUM**: Tx FIFO number  
These bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.  
This field is valid only for IN endpoints.
- Bit 21 **STALL**: STALL handshake  
Applies to non-control, non-isochronous IN endpoints only (access type is rw).  
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.  
  
Applies to control endpoints only (access type is rs).  
The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.
- Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP[1:0]**: Endpoint type

This is the transfer type supported by this logical endpoint.

- 00: Control
- 01: Isochronous
- 10: Bulk
- 11: Interrupt

Bit 17 **NAKSTS**: NAK status

It indicates the following:

- 0: The core is transmitting non-NAK handshakes based on the FIFO status.
- 1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there are data available in the Tx FIFO.

For isochronous IN endpoints: The core sends out a zero-length data packet, even if there are data available in the Tx FIFO.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM**: Even/odd frame

Applies to isochronous IN endpoints only.

Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.

- 0: Even frame
- 1: Odd frame

**DPID**: Endpoint data PID

Applies to interrupt/bulk IN endpoints only.

Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.

- 0: DATA0
- 1: DATA1

Bit 15 **USBAEP**: USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]**: Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

### 50.15.44 OTG device IN endpoint x interrupt register (OTG\_DIEPINTx) (x = 0..5, where x = Endpoint number)

Address offset: 0x908 + (x \* 0x20)

Reset value: 0x0000 0080

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 588*. The application must read this register when the IN endpoints interrupt bit of the core interrupt register (IEPINT in OTG\_GINTSTS) is set. Before the application can read this register, it must first read the device all endpoints interrupt (OTG\_DAINTE) register to get the exact endpoint number for the device endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG\_DAINTE and OTG\_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAK	Res.	PKTD RPSTS	Res.	Res.	TXFIF OUD RN	TXFE	IN EPNE	IN EPNM	ITTXFE	TOC	Res.	EP DISD	XFRC
		rc_w1		rc_w1			rc_w1	r	r	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAK**: NAK input

The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **PKTDRPSTS**: Packet dropped status

This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.

Bit 10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **TXFIFOURN**: Transmit Fifo Underrun (TxfifoUndrn)

The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. Dependency: This interrupt is valid only when Thresholding is enabled

Bit 7 **TXFE**: Transmit FIFO empty

This interrupt is asserted when the Tx FIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the Tx FIFO Empty Level bit in the OTG\_GAHBCFG register (TXFELVL bit in OTG\_GAHBCFG).

- Bit 6 **INEPNE**: IN endpoint NAK effective  
 This bit can be cleared when the application clears the IN endpoint NAK by writing to the CNAK bit in OTG\_DIEPCTLx.  
 This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core.  
 This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.
- Bit 5 **INEPNM**: IN token received with EP mismatch  
 Indicates that the data in the top of the non-periodic Tx FIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received.
- Bit 4 **ITTXFE**: IN token received when Tx FIFO is empty  
 Indicates that an IN token was received when the associated Tx FIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.
- Bit 3 **TOC**: Timeout condition  
 Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.
- Bit 2 Reserved, must be kept at reset value
- Bit 1 **EPDISD**: Endpoint disabled interrupt  
 This bit indicates that the endpoint is disabled per the application's request.
- Bit 0 **XFRC**: Transfer completed interrupt  
 This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

**50.15.45 OTG device IN endpoint 0 transfer size register (OTG\_DIEPTSIZ0)**

Address offset: 0x910

Reset value: 0x0000 0000

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the device control endpoint 0 control registers (EPENA in OTG\_DIEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT[1:0]		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.	Res.	Res.	Res.	Res.	Res.	XFRSIZ[6:0]								
									rw	rw	rw	rw	rw	rw	rw		



Bits 31:21 Reserved, must be kept at reset value.

Bits 20:19 **PKTCNT[1:0]**: Packet count

Indicates the total number of USB packets that constitute the transfer size amount of data for endpoint 0.

This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ[6:0]**: Transfer size

Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

### 50.15.46 OTG device IN endpoint transmit FIFO status register (OTG\_DTXFSTSx) (x = 0..5, where x = endpoint number)

Address offset for IN endpoints: 0x918 + (x \* 0x20) This read-only register contains the free space information for the device IN endpoint Tx FIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTFSAV[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 **INEPTFSAV[15:0]**: IN endpoint Tx FIFO space available

Indicates the amount of free space available in the endpoint Tx FIFO.

Values are in terms of 32-bit words:

0x0: Endpoint Tx FIFO is full

0x1: 1 word available

0x2: 2 words available

0xn: n words available

Others: Reserved

**50.15.47 OTG device IN endpoint x transfer size register (OTG\_DIEPTSIZx)  
(x = 1..5, where x = endpoint number)**

Address offset: 0x910 + (x \* 0x20)

Reset value: 0x0000 0000

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using the endpoint enable bit in the OTG\_DIEPCTLx registers (EPENA bit in OTG\_DIEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCNT[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XFRSIZ[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **MCNT[1:0]**: Multi count

For periodic IN endpoints, this field indicates the number of packets that must be transmitted per frame on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints.

- 01: 1 packet
- 10: 2 packets
- 11: 3 packets

Bits 28:19 **PKTCNT[9:0]**: Packet count

Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.

This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:0 **XFRSIZ[18:0]**: Transfer size

This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

### 50.15.48 OTG device control OUT endpoint 0 control register (OTG\_DOEPCTL0)

Address offset: 0xB00

Reset value: 0x0000 8000

This section describes the OTG\_DOEPCTL0 register. Nonzero control endpoints use registers for endpoints 1–3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	Res.	Res.	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP[1:0]		NAK STS	Res.
w	r			w	w					rs	rw	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ[1:0]	
r														r	r

Bit 31 **EPENA**: Endpoint enable

The application sets this bit to start transmitting data on endpoint 0.

The core clears this bit before setting any of the following interrupts on this endpoint:

- SETUP phase done
- Endpoint disabled
- Transfer completed

Bit 30 **EPDIS**: Endpoint disable

The application cannot disable control OUT endpoint 0.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **SNAK**: Set NAK

A write to this bit sets the NAK bit for the endpoint.

Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK**: Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL**: STALL handshake

The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM**: Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]**: Endpoint type

Hardcoded to 2'b00 for control.

Bit 17 **NAKSTS**: NAK status

Indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit, the core stops receiving data, even if there is space in the Rx FIFO to accommodate the incoming packet. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **USBAEP**: USB active endpoint

This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.

Bits 14:2 Reserved, must be kept at reset value.

Bits 1:0 **MPSIZ[1:0]**: Maximum packet size

The maximum packet size for control OUT endpoint 0 is the same as what is programmed in control IN endpoint 0.

00: 64 bytes

01: 32 bytes

10: 16 bytes

11: 8 bytes

### 50.15.49 OTG device OUT endpoint x interrupt register (OTG\_DOEPINTx) (x = 0..5, where x = Endpoint number)

Address offset: 0xB08 + (x \* 0x20)

Reset value: 0x0000 0080

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in [Figure 588](#). The application must read this register when the OUT endpoints interrupt bit of the OTG\_GINTSTS register (OEPINT bit in OTG\_GINTSTS) is set. Before the application can read this register, it must first read the OTG\_DAINTR register to get the exact endpoint number for the OTG\_DOEPINTx register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG\_DAINTR and OTG\_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NYET	NAK	BERR	Res.	Res.	Res.	OUT PKT ERR	Res.	Res.	STSPH SRX	OEP DIS	STUP	Res.	EP DISD	XFRC
	rc_w1	rc_w1	rc_w1				rc_w1			rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **NYET**: NYET interrupt

This interrupt is generated when a NYET response is transmitted for a non isochronous OUT endpoint.



- Bit 13 **NAK**: NAK input  
The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.
- Bit 12 **BERR**: Babble error interrupt  
The core generates this interrupt when babble is received for the endpoint.
- Bits 11:10 Reserved, must be kept at reset value.
- Bit 9 Reserved, must be kept at reset value.
- Bit 8 **OUTPKTERR**: OUT packet error  
This interrupt is asserted when the core detects an overflow or a CRC error for an OUT packet. This interrupt is valid only when thresholding is enabled.
- Bit 7 Reserved, must be kept at reset value.
- Bit 6 Reserved, must be kept at reset value.
- Bit 5 **STSPHSRX**: Status phase received for control write  
This interrupt is valid only for control OUT endpoints. This interrupt is generated only after OTG\_FS has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a control write transfer. The application can use this interrupt to ACK or STALL the status phase, after it has decoded the data phase.
- Bit 4 **OTEPDIS**: OUT token received when endpoint disabled  
Applies only to control OUT endpoints.  
Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.
- Bit 3 **STUP**: SETUP phase done  
Applies to control OUT endpoint only.  
Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.
- Bit 2 Reserved, must be kept at reset value.
- Bit 1 **EPDISD**: Endpoint disabled interrupt  
This bit indicates that the endpoint is disabled per the application's request.
- Bit 0 **XFRC**: Transfer completed interrupt  
This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

**50.15.50 OTG device OUT endpoint 0 transfer size register (OTG\_DOEPTSIZ0)**

Address offset: 0xB10

Reset value: 0x0000 0000

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the OTG\_DOEPCTL0 registers (EPENA bit in OTG\_DOEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–5.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	STUPCNT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT	Res.	Res.	Res.
	rw	rw										rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **STUPCNT[1:0]**: SETUP packet count

This field specifies the number of back-to-back SETUP data packets the endpoint can receive.

- 01: 1 packet
- 10: 2 packets
- 11: 3 packets

Bits 28:20 Reserved, must be kept at reset value.

Bit 19 **PKTCNT**: Packet count

This field is decremented to zero after a packet is written into the Rx FIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ[6:0]**: Transfer size

Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.

**50.15.51 OTG device OUT endpoint x control register (OTG\_DOEPTLx)  
(x = 1..5, where x = endpoint number)**

Address offset for OUT endpoints: 0xB00 + (x \* 0x20)

Reset value: 0x0000 0000

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SD1 PID/ SODD FRM	SD0 PID/ SEVN FRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP[1:0]		NAK STS	EO NUM/ DPID
rs	rs	w	w	w	w					rw/rs	rw	rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

**Bit 31 EPENA:** Endpoint enable

Applies to IN and OUT endpoints.  
 The application sets this bit to start transmitting data on an endpoint.  
 The core clears this bit before setting any of the following interrupts on this endpoint:

- SETUP phase done
- Endpoint disabled
- Transfer completed

**Bit 30 EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

**Bit 29 SD1PID:** Set DATA1 PID

Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the endpoint data PID (DPID) field in this register to DATA1.

**SODDFRM:** Set odd frame

Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

**Bit 28 SD0PID:** Set DATA0 PID

Applies to interrupt/bulk OUT endpoints only.  
 Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

**SEVNFRM:** Set even frame

Applies to isochronous OUT endpoints only.  
 Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

**Bit 27 SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.  
 Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.



**Bit 26 CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

**Bit 21 STALL:** STALL handshake

Applies to non-control, non-isochronous OUT endpoints only (access type is rw).

The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Applies to control endpoints only (access type is rs).

The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

**Bit 20 SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

**Bits 19:18 EPTYP[1:0]:** Endpoint type

This is the transfer type supported by this logical endpoint.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

**Bit 17 NAKSTS:** NAK status

Indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

The core stops receiving any data on an OUT endpoint, even if there is space in the Rx FIFO to accommodate the incoming packet.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

**Bit 16 EONUM:** Even/odd frame

Applies to isochronous IN and OUT endpoints only.

Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.

0: Even frame

1: Odd frame

**DPID:** Endpoint data PID

Applies to interrupt/bulk OUT endpoints only.

Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.

0: DATA0

1: DATA1

Bit 15 **USBAEP**: USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]**: Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

**50.15.52 OTG device OUT endpoint x transfer size register (OTG\_DOEPTSIZE<sub>x</sub>) (x = 1..5, where x = Endpoint number)**

Address offset: 0xB10 + (x \* 0x20)

Reset value: 0x0000 0000

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using endpoint enable bit of the OTG\_DOEPTLx registers (EPENA bit in OTG\_DOEPTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RXDPID/ STUPCNT[1:0]		PKTCNT[9:0]										XFRSIZ		
	r/rw	r/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
XFRSIZ															
rw	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 30:29 **RXDPID[1:0]**: Received data PID

Applies to isochronous OUT endpoints only.

This is the data PID received in the last packet for this endpoint.

00: DATA0

10: DATA1

**STUPCNT[1:0]: SETUP packet count**

Applies to control OUT endpoints only.

This field specifies the number of back-to-back SETUP data packets the endpoint can receive.

01: 1 packet

10: 2 packets

11: 3 packets

Bits 28:19 **PKTCNT[9:0]: Packet count**

Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.

This field is decremented every time a packet (maximum size or short packet) is written to the Rx FIFO.

Bits 18:0 **XFRSIZ: Transfer size**

This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.

**50.15.53 OTG power and clock gating control register (OTG\_PCGCCTL)**

Address offset: 0xE00

Reset value: 0x200B 8000

This register is available in host and device 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	PHY SLEEP	ENL1 GTG	PHY SUSP	Res.	Res.	GATE HCLK	STPP CLK
								r	r	rw	r			rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **SUSP: Deep Sleep**

This bit indicates that the PHY is in Deep Sleep when in L1 state.

Bit 6 **PHYSLEEP: PHY in Sleep**

This bit indicates that the PHY is in the Sleep state.

Bit 5 **ENL1GTG: Enable sleep clock gating**

When this bit is set, core internal clock gating is enabled in Sleep state if the core cannot assert utmi\_i1\_suspend\_n. When this bit is not set, the PHY clock is not gated in Sleep state.

Bit 4 **PHYSUSP: PHY suspended**

Indicates that the PHY has been suspended. This bit is updated once the PHY is suspended after the application has set the STPPCLK bit.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **GATEHCLK**: Gate HCLK

The application sets this bit to gate HCLK to modules other than the AHB Slave and Master and wakeup logic when the USB is suspended or the session is not valid. The application clears this bit when the USB is resumed or a new session starts.

Bit 0 **STPPCLK**: Stop PHY clock

The application sets this bit to stop the PHY clock when the USB is suspended, the session is not valid, or the device is disconnected. The application clears this bit when the USB is resumed or a new session starts.

### 50.15.54 OTG\_FS register map

The table below gives the USB OTG register map and reset values.

**Table 329. OTG\_FS 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	<b>OTG_GOTGCTL</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CURMOD	OTGVER	BSVLD	ASVLD	DBCT	CIDSTS	Res.	Res.	Res.	EHEN	DHNPEN	HSHNPEN	HNPRQ	HNGSCS	BVALOVAL	BVALOEN	AVALOVAL	AVALOEN	VBVALOVA	VBVALOEN	SRQ	SRQSCS		
	Reset value												0	0	0	0	0	1				0	0	0	0	0	0	0	0	0	0	0	0	0		
0x004	<b>OTG_GOTGINT</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDCHNG	DBCDNE	ADTOCHG	HNGDET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HNSSCHG	SRSSCHG	Res.	Res.	Res.	Res.	SEDET	Res.	Res.			
	Reset value												0	0	0	0									0	0					0					
0x008	<b>OTG_GAHBCFG</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTXFELVL	TXFELVL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GINTMSK		
	Reset value																								0	0									0	
0x00C	<b>OTG_GUSBCFG</b>	Res.	FDMOD	FHMOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRDT	
	Reset value		0	0																					0	0	0	1								0
0x010	<b>OTG_GRSTCTL</b>	AHBIDL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFNUM
	Reset value	1																																		0
0x014	<b>OTG_GINTSTS</b>	WKUPINT	SRQINT	DISCINT	CIDSCHG	LPMINT	PTXFE	HCINT	HPRTINT	RSTDET	Res.	IPXFR/INCOMPISOOUT	IISOXFR	OEPINT	IEPINT	Res.	Res.	EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Res.	Res.	GONAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD	Res.		
	Reset value	0	0	0	1	0	1	0	0	0		0	0	0	0			0	0	0	0	0	0			0	0	1	0	0	0	0	0	0		



Table 329. OTG\_FS 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		
0x018	<b>OTG_GINTMSK</b>	WUIM	SRQIM	DISCINT	CIDSCHGM	LPINTM	PTXFEM	HCIM	PRTIM	RSTDETM	Res.	IPXFRM/IISOXFRM	IISOXFRM	OEPIINT	IEPIINT	Res.	Res.	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Res.	Res.	GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	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		
0x01C	<b>OTG_GRXSTSR (host mode)</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS				DPID	BCNT								CHNUM									
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	<b>OTG_GRXSTSR (Device mode)</b>	Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM				PKTSTS				DPID	BCNT								EPNUM									
	Reset value					0			0	0	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	<b>OTG_GRXSTSP (host mode)</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS				DPID	BCNT								CHNUM									
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	<b>OTG_GRXSTSP (Device mode)</b>	Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM				PKTSTS				DPID	BCNT								EPNUM									
	Reset value					0			0	0	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	<b>OTG_GRXFSIZ</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFD																	
	Reset value																	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0			
0x028	<b>OTG_HNPTXFSIZ/ OTG_DIEPTXF0</b>	NPTXFD/TX0FD										NPTXFSA/TX0FSA																							
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0		
0x02C	<b>OTG_HNPTXSTS</b>	Res.	NPTXQTOP						NPTQXSAV						NPTXFSAV																				
	Reset value		0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0			
0x038	<b>OTG_GCCFG</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBDEN	SDEN	PDEN	DCDEN	BCDEN	PWRDWN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2DET	SDET	PDET	DCDET
	Reset value												0	0	0	0	0	0													X	X	X	X	
0x03C	<b>OTG_CID</b>	PRODUCT_ID																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0		
0x054	<b>OTG_GLPMCFG</b>	Res.	Res.	Res.	ENBESL	LPMR CNTSTS			SNDLPM	LPM RCNT		LPMCHIDX				L1RSMOK	SLPSTS	LPM RSP		L1DSEN	BESLTHRS				L1SSEN	REMWAKE	BESL				LPMACK	LPMEN			
	Reset value				0	0	0	0	0	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 329. OTG\_FS 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	
0x058	OTG_GPWRDN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADPIF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADPMEN	
	Reset value									0																								0
0x060	OTG_GADPCTL	Res.	Res.	Res.	AR		ADPTOIM	ADPSNSIM	ADPPRBIM	ADPTOIF	ADPSNSIF	ADPPRBIF	ADPEN	ADPRST	ENASNS	ENAPRB																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	OTG_HPTXFSIZ	PTXFSIZ										PTXSA																						
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
0x104	OTG_DIEPTXF1	INEPTXFD										INEPTXSA																						
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
0x108	OTG_DIEPTXF2	INEPTXFD										INEPTXSA																						
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0		
0x114	OTG_DIEPTXF5	INEPTXFD										INEPTXSA																						
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0		
0x400	OTG_HCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSLSS	FSLPCS	
	Reset value																														0	0	0	
0x404	OTG_HFIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLDCTR																	
	Reset value																0	1	1	1	0	1	0	1	0	0	1	1	1	0	0	0	0	
0x408	OTG_HFNUM	FTREM										FRNUM																						
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1		
0x410	OTG_HPTXSTS	PTXQTOP					PTXQSAV					PTXFSAVL																						
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0		
0x414	OTG_HAINT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																	
	Reset value																																	
0x418	OTG_HAINTMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																	
	Reset value																																	

Table 329. OTG\_FS 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	
0x440	OTG_HPRT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	
0x500	OTG_HCCHAR0	CHENA	CHDIS	ODDFRM	DAD						MCNT	EPTYP	LSDEV	Res.	EPDIR	EPNUM				MPSIZ														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x508	OTG_HCINT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRM
	Reset value																							0	0	0	0	0	0	0	0	0	0	0
0x50C	OTG_HCINTMSK0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMORM	BBERRM	TXERRM	Res.	ACKM	NAKM	STALLM	Res.	CHHM	XFRM
	Reset value																							0	0	0	0	0	0	0	0	0	0	0
0x510	OTG_HCTSIZ0	Res.	DPID		PKTCNT								XFRSIZ																					
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x520	OTG_HCCHAR1	CHENA	CHDIS	ODDFRM	DAD						MCNT	EPTYP	LSDEV	Res.	EPDIR	EPNUM				MPSIZ														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x528	OTG_HCINT1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRM
	Reset value																							0	0	0	0	0	0	0	0	0	0	0
0x52C	OTG_HCINTMSK1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	Res.	ACKM	NAKM	STALLM	Res.	CHHM	XFRM
	Reset value																							0	0	0	0	0	0	0	0	0	0	0
0x530	OTG_HCTSIZ1	Res.	DPID		PKTCNT								XFRSIZ																					
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...																																	
0x660	OTG_HCCHAR11	CHENA	CHDIS	ODDFRM	DAD						MCNT	EPTYP	LSDEV	Res.	EPDIR	EPNUM				MPSIZ														
	Reset value	0	0	0	0	0	0	0	0	0	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 329. OTG\_FS 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		
0x668	OTG_HCINT11	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC		
	Reset value																						0	0	0	0	0	0	0	0	0	0	0		
0x66C	OTG_HCINTMSK11	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	Res.	ACKM	NAKM	STALLM	Res.	CHHM	XFRM		
	Reset value																						0	0	0	0	0	0	0	0	0	0	0		
0x670	OTG_HCTSIZ11	Res.	DPID	PKTCNT											XFRSIZ																				
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x800	OTG_DCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRATIM	XCVRDLY	Res.	PFIVL	DAD											Res.	NZLSOHSK	DSPD
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x804	OTG_DCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSBESLJCT	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		
0x808	OTG_DSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																		
0x810	OTG_DIEPMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																		
0x814	OTG_DOEPMASK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																		
0x818	OTG_DAIN1	OEPINT														IEPINT																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x81C	OTG_DAIN1MSK	OEP1M														IEP1M																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x828	OTG_DVBUSDIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 329. OTG\_FS 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			
0x82C	OTG_DVB_USPULSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DVBUSP																		
	Reset value																	0	0	0	0	0	0	1	0	1	1	0	1	1	1	0	0	0		
0x834	OTG_DIE_PEMPMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTXFEM																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x900	OTG_DIEPCTL0	EPENA	EPDIS	Res.	Res.	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP	Res.	NAKSTS	Res.	USBAEP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ		
	Reset value	0	0			0	0	0	0	0	0	0		0	0	0		1																0	0	
0x908	OTG_DIEPINT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	Res.	EPDIS	XFRC		
	Reset value																									0	1	0	0	0	0	0	0	0	0	
0x910	OTG_DIEPTSIZ0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKT CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ	
	Reset value													0	0																				0	0
0x918	OTG_DTXFSTS0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV																		
	Reset value																	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
0x920	OTG_DIEPCTL1	EPENA	EPDIS	SODDFRM/SDIPID	SD0PID/SEVNFIRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP	Res.	NAKSTS	Res.	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ	
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0																	0	0
0x928	OTG_DIEPINT1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	Res.	EPDIS	XFRC		
	Reset value																									0	1	0	0	0	0	0	0	0	0	0
0x930	OTG_DIEPTSIZ1	Res.	MCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x938	OTG_DTXFSTS1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV																		
	Reset value																	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
0x940	OTG_DIEPCTL2	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFIRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP	Res.	NAKSTS	Res.	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0																	0	0



Table 329. OTG\_FS 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		
...	...																																		
0x9A0	<b>OTG_DIEPCTL5</b>	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFIRM	SNAK	CNAK	TXFNUM					STALL	Res.	EPTYP	NAKSTS			EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
...	...																																		
0x9A8	<b>OTG_DIEPINT5</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.	TXFIFOUDRN	TXFE	INEPNE	INENPM	ITTXFE	TOC	Res.	EPDIS	XFRC	
	Reset value																									0	1	0	0	0	0	0	0	0	0
...	...																																		
0x9B8	<b>OTG_DTXFSTS5</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV																	
	Reset value																	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
...	...																																		
0x9B0	<b>OTG_DIEPTSIZ5</b>	Res.	MCNT	PKTCNT											XFRSIZ																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0xB00	<b>OTG_DOEPCCTL0</b>	EPENA	EPDIS	Res.	Res.	SNAK	CNAK	Res.	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP	NAKSTS			USBAEP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ	
	Reset value	0	0			0	0						0	0	0	0	0	0	1														0	0	
0xB08	<b>OTG_DOEPIINT0</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NYET	NAK	BERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																			0	0	0													
0xB10	<b>OTG_DOEPTSIZ0</b>	Res.	STUPCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																				



Table 329. OTG\_FS 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				
0xB20	OTG_DOEPC1	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFMRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP	NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0xB28	OTG_DOEPINT1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NYET	NAK	BERR	Res.	Res.	Res.	OUTPKTERR	Res.	Res.	STSPHSRX	OTEPDIS	STUP	Res.	EPDIS	XFRC				
	Reset value																		0	0	0				0			0	0	0	0	0	0				
0xB30	OTG_DOEPTSIZ1	Res.	RXDPID/ STUPCNT	PKTCNT								XFRSIZ																									
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
...	...																																				
0xBA0	OTG_DOEPC5	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFMRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP	NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0xBA8	OTG_DOEPINT5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NYET	NAK	BERR	Res.	Res.	Res.	OUTPKTERR	Res.	Res.	STSPHSRX	OTEPDIS	STUP	Res.	EPDIS	XFRC				
	Reset value																		0	0	0				0			0	0	0	0	0	0				
0xBB0	OTG_DOEPTSIZ5	Res.	RXDPID/ STUPCNT	PKTCNT								XFRSIZ																									
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0xE00	OTG_PCGCCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	PHYSLEEP	ENL1GTG	PHYSUSP	Res.	Res.	GATECLK	STPPCLK				
	Reset value																									0	0	0	0			0	0				

Refer to [Section 2.2.2 on page 82](#) for the register boundary addresses.



## 50.16 OTG\_FS programming model

### 50.16.1 Core initialization

The application must perform the core initialization sequence. If the cable is connected during power-up, the current mode of operation bit in the OTG\_GINTSTS (CMOD bit in OTG\_GINTSTS) reflects the mode. The OTG\_FS controller enters host mode when an “A” plug is connected or device mode when a “B” plug is connected.

This section explains the initialization of the OTG\_FS controller after power-on. The application must follow the initialization sequence irrespective of host or device mode operation. All core global registers are initialized according to the core's configuration:

1. Program the following fields in the OTG\_GAHBCFG register:
  - Global interrupt mask bit GINTMSK = 1
  - Rx FIFO non-empty (RXFLVL bit in OTG\_GINTSTS)
  - Periodic Tx FIFO empty level
2. Program the following fields in the OTG\_GUSBCFG register:
  - HNP capable bit
  - SRP capable bit
  - OTG\_FS timeout calibration field
  - USB turnaround time field
3. The software must unmask the following bits in the OTG\_GINTMSK register:
  - OTG interrupt mask
  - Mode mismatch interrupt mask
4. The software can read the CMOD bit in OTG\_GINTSTS to determine whether the OTG\_FS controller is operating in host or device mode.

### 50.16.2 Host initialization

To initialize the core as host, the application must perform the following steps:

1. Program the HPRTINT in the OTG\_GINTMSK register to unmask
2. Program the OTG\_HCFG register to select full-speed host
3. Program the PPWR bit in OTG\_HPRT to 1. This drives  $V_{BUS}$  on the USB.
4. Wait for the PCDET interrupt in OTG\_HPRT0. This indicates that a device is connecting to the port.
5. Program the PRST bit in OTG\_HPRT to 1. This starts the reset process.
6. Wait at least 10 ms for the reset process to complete.
7. Program the PRST bit in OTG\_HPRT to 0.
8. Wait for the PENCHNG interrupt in OTG\_HPRT.
9. Read the PSPD bit in OTG\_HPRT to get the enumerated speed.
10. Program the HFIR register with a value corresponding to the selected PHY clock 1
11. Program the FSLSPCS field in the OTG\_HCFG register following the speed of the device detected in step 9. If FSLSPCS has been changed a port reset must be performed.
12. Program the OTG\_GRXFSIZ register to select the size of the receive FIFO.
13. Program the OTG\_HNPTXFSIZ register to select the size and the start address of the Non-periodic transmit FIFO for non-periodic transactions.
14. Program the OTG\_HPTXFSIZ register to select the size and start address of the periodic transmit FIFO for periodic transactions.

To communicate with devices, the system software must initialize and enable at least one channel.

### 50.16.3 Device initialization

The application must perform the following steps to initialize the core as a device on power-up or after a mode change from host to device.

1. Program the following fields in the OTG\_DCFG register:
  - Device speed
  - Non-zero-length status OUT handshake
2. Program the OTG\_GINTMSK register to unmask the following interrupts:
  - USB reset
  - Enumeration done
  - Early suspend
  - USB suspend
  - SOF
3. Wait for the USBRST interrupt in OTG\_GINTSTS. It indicates that a reset has been detected on the USB that lasts for about 10 ms on receiving this interrupt.

Wait for the ENUMDNE interrupt in OTG\_GINTSTS. This interrupt indicates the end of reset on the USB. On receiving this interrupt, the application must read the OTG\_DSTS register to determine the enumeration speed and perform the steps listed in [Endpoint initialization on enumeration completion on page 1993](#).

At this point, the device is ready to accept SOF packets and perform control transfers on control endpoint 0.



## 50.16.4 Host programming model

### Channel initialization

The application must initialize one or more channels before it can communicate with connected devices. To initialize and enable a channel, the application must perform the following steps:

1. Program the OTG\_GINTMSK register to unmask the following:
2. Channel interrupt
  - Non-periodic transmit FIFO empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
  - Non-periodic transmit FIFO half-empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
3. Program the OTG\_HAINTMSK register to unmask the selected channels' interrupts.
4. Program the OTG\_HCINTMSK register to unmask the transaction-related interrupts of interest given in the host channel interrupt register.
5. Program the selected channel's OTG\_HCTSIZx register with the total transfer size, in bytes, and the expected number of packets, including short packets. The application must program the PID field with the initial data PID (to be used on the first OUT transaction or to be expected from the first IN transaction).
6. Program the OTG\_HCCHARx register of the selected channel with the device's endpoint characteristics, such as type, speed, direction, and so forth. (The channel can be enabled by setting the channel enable bit to 1 only when the application is ready to transmit or receive any packet).

### Halting a channel

The application can disable any channel by programming the OTG\_HCCHARx register with the CHDIS and CHENA bits set to 1. This enables the OTG\_FS host to flush the posted requests (if any) and generates a channel halted interrupt. The application must wait for the CHH interrupt in OTG\_HCINTx before reallocating the channel for other transactions. The OTG\_FS host does not interrupt the transaction that has already been started on the USB.

Before disabling a channel, the application must ensure that there is at least one free space available in the non-periodic request queue (when disabling a non-periodic channel) or the periodic request queue (when disabling a periodic channel). The application can simply flush the posted requests when the request queue is full (before disabling the channel), by programming the OTG\_HCCHARx register with the CHDIS bit set to 1, and the CHENA bit cleared to 0.

The application is expected to disable a channel on any of the following conditions:

1. When an STALL, TXERR, BBERR or DTERR interrupt in OTG\_HCINTx is received for an IN or OUT channel. The application must be able to receive other interrupts (DTERR, Nak, data, TXERR) for the same channel before receiving the halt.
2. When a DISCINT (disconnect device) interrupt in OTG\_GINTSTS is received. (The application is expected to disable all enabled channels).
3. When the application aborts a transfer before normal completion.

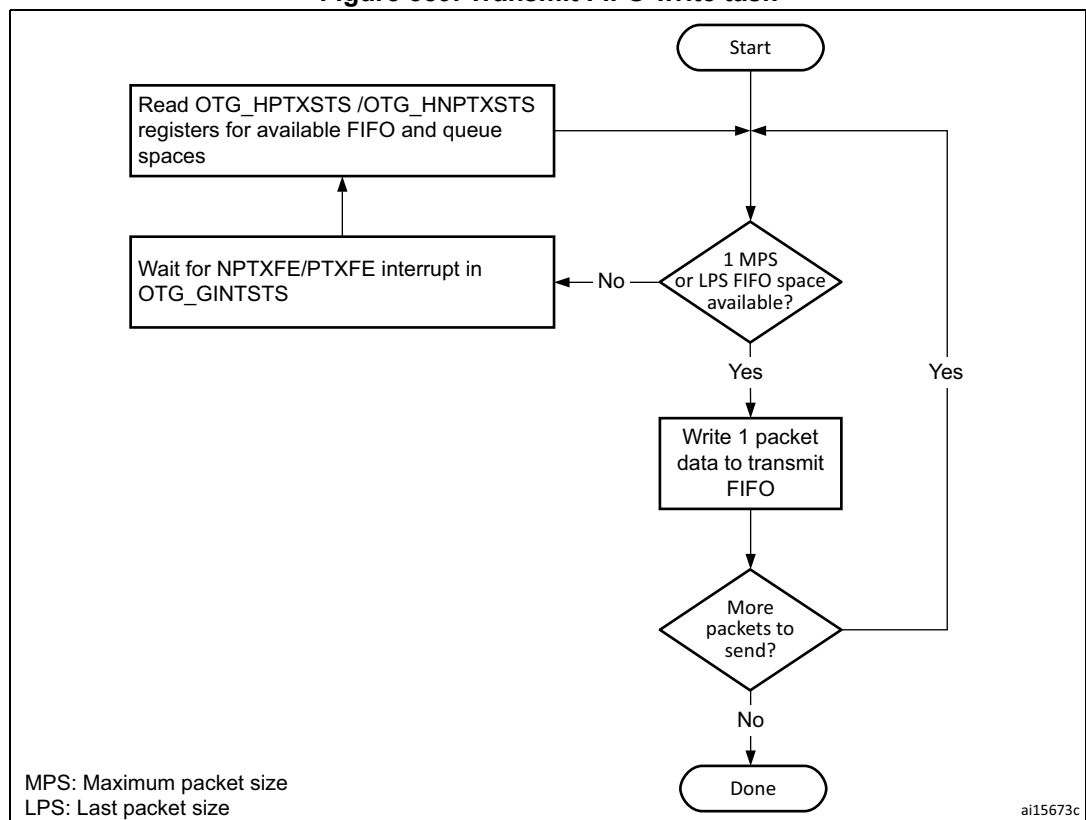
**Operational model**

The application must initialize a channel before communicating to the connected device. This section explains the sequence of operation to be performed for different types of USB transactions.

- **Writing the transmit FIFO**

The OTG\_FS host automatically writes an entry (OUT request) to the periodic/non-periodic request queue, along with the last 32-bit word write of a packet. The application must ensure that at least one free space is available in the periodic/non-periodic request queue before starting to write to the transmit FIFO. The application must always write to the transmit FIFO in 32-bit words. If the packet size is non-32-bit word aligned, the application must use padding. The OTG\_FS host determines the actual packet size based on the programmed maximum packet size and transfer size.

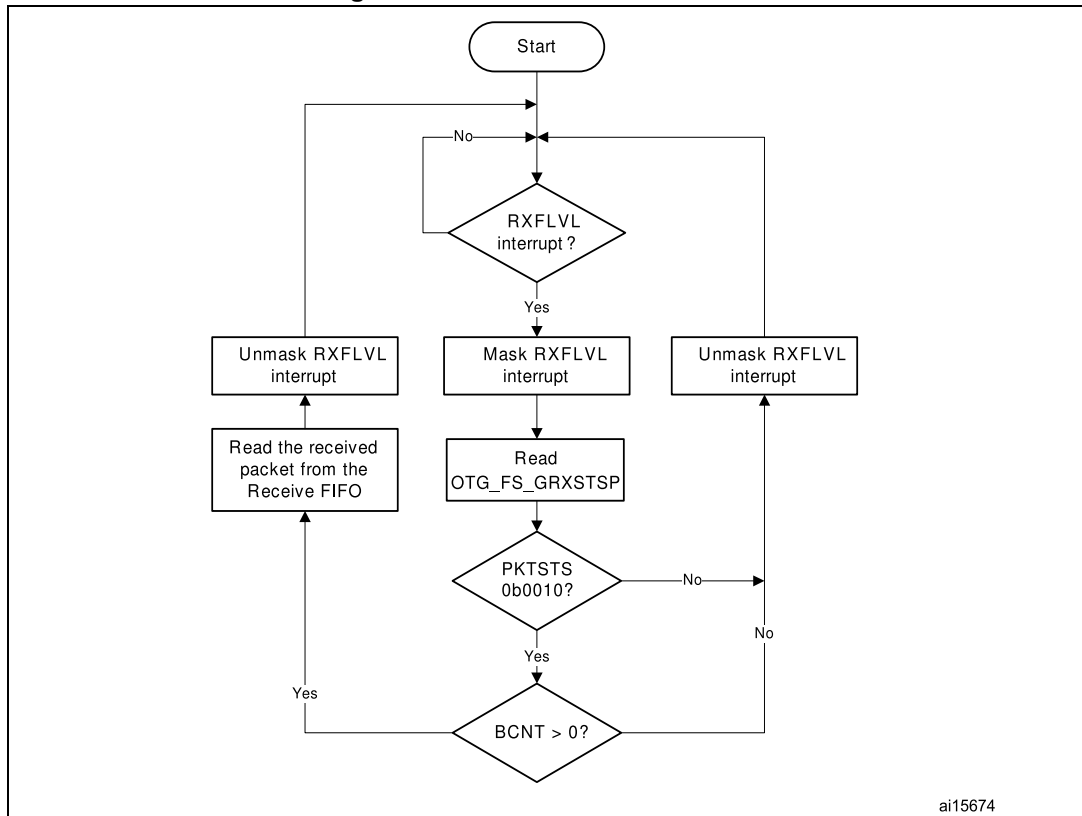
**Figure 589. Transmit FIFO write task**



- **Reading the receive FIFO**

The application must ignore all packet statuses other than IN data packet (bx0010).

Figure 590. Receive FIFO read task



ai15674

• **Bulk and control OUT/SETUP transactions**

A typical bulk or control OUT/SETUP pipelined transaction-level operation is shown in [Figure 591](#). See channel 1 (ch\_1). Two bulk OUT packets are transmitted. A control SETUP transaction operates in the same way but has only one packet. The assumptions are:

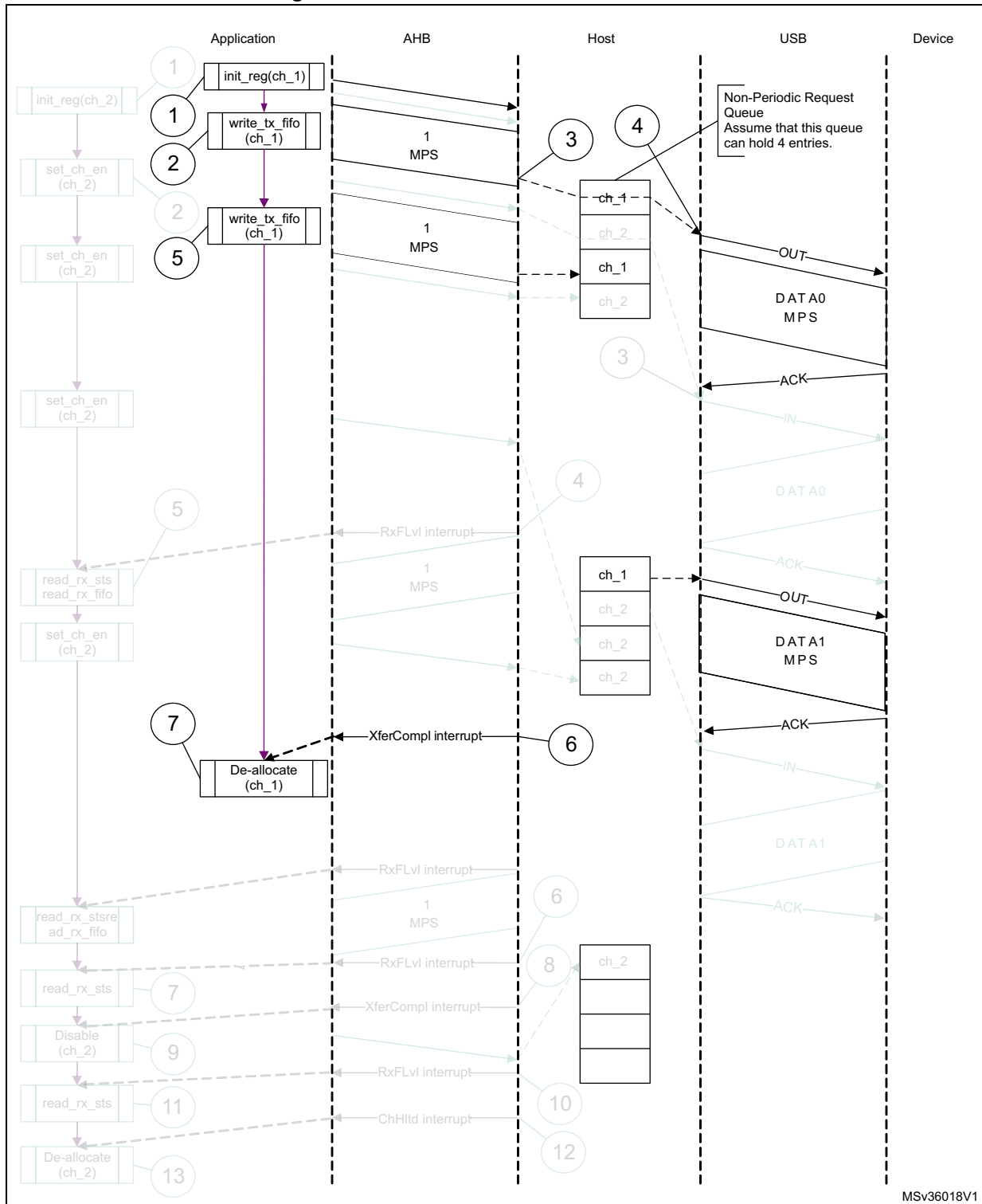
- The application is attempting to send two maximum-packet-size packets (transfer size = 1,024 bytes).
- The non-periodic transmit FIFO can hold two packets (128 bytes for FS).
- The non-periodic request queue depth = 4.

• **Normal bulk and control OUT/SETUP operations**

The sequence of operations in (channel 1) is as follows:

1. Initialize channel 1
2. Write the first packet for channel 1
3. Along with the last word write, the core writes an entry to the non-periodic request queue
4. As soon as the non-periodic queue becomes non-empty, the core attempts to send an OUT token in the current frame
5. Write the second (last) packet for channel 1
6. The core generates the XFRC interrupt as soon as the last transaction is completed successfully
7. In response to the XFRC interrupt, de-allocate the channel for other transfers
8. Handling non-ACK responses

Figure 591. Normal bulk/control OUT/SETUP



1. The grayed elements are not relevant in the context of this figure.

The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions is shown in the following code samples.

- **Interrupt service routine for bulk/control OUT/SETUP and bulk/control IN transactions**

- a) Bulk/control OUT/SETUP

```

Unmask (NAK/TXERR/STALL/XFRC)
if (XFRC)
{
    Reset Error Count
    Mask ACK
    De-allocate Channel
}
else if (STALL)
{
    Transfer Done = 1
    Unmask CHH
    Disable Channel
}
else if (NAK or TXERR )
{
    Rewind Buffer Pointers
    Unmask CHH
    Disable Channel
    if (TXERR)
    {
        Increment Error Count
        Unmask ACK
    }
    else
    {
        Reset Error Count
    }
}
else if (CHH)
{
    Mask CHH
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else
    {
        Re-initialize Channel
    }
}

```

```

else if (ACK)
{
  Reset Error Count
  Mask ACK
}

```

The application is expected to write the data packets into the transmit FIFO when the space is available in the transmit FIFO and the request queue. The application can make use of the NPTXFE interrupt in OTG\_GINTSTS to find the transmit FIFO space.

#### b) Bulk/control IN

```

Unmask (TXERR/XFRC/BBERR/STALL/DTERR)
if (XFRC)
{
  Reset Error Count
  Unmask CHH
  Disable Channel
  Reset Error Count
  Mask ACK
}
else if (TXERR or BBERR or STALL)
{
  Unmask CHH
  Disable Channel
  if (TXERR)
  {
    Increment Error Count
    Unmask ACK
  }
}
else if (CHH)
{
  Mask CHH
  if (Transfer Done or (Error_count == 3))
  {
    De-allocate Channel
  }
  else
  {
    Re-initialize Channel
  }
}
else if (ACK)
{
  Reset Error Count
  Mask ACK
}

```

```
else if (DTERR)
{
    Reset Error Count
}
```

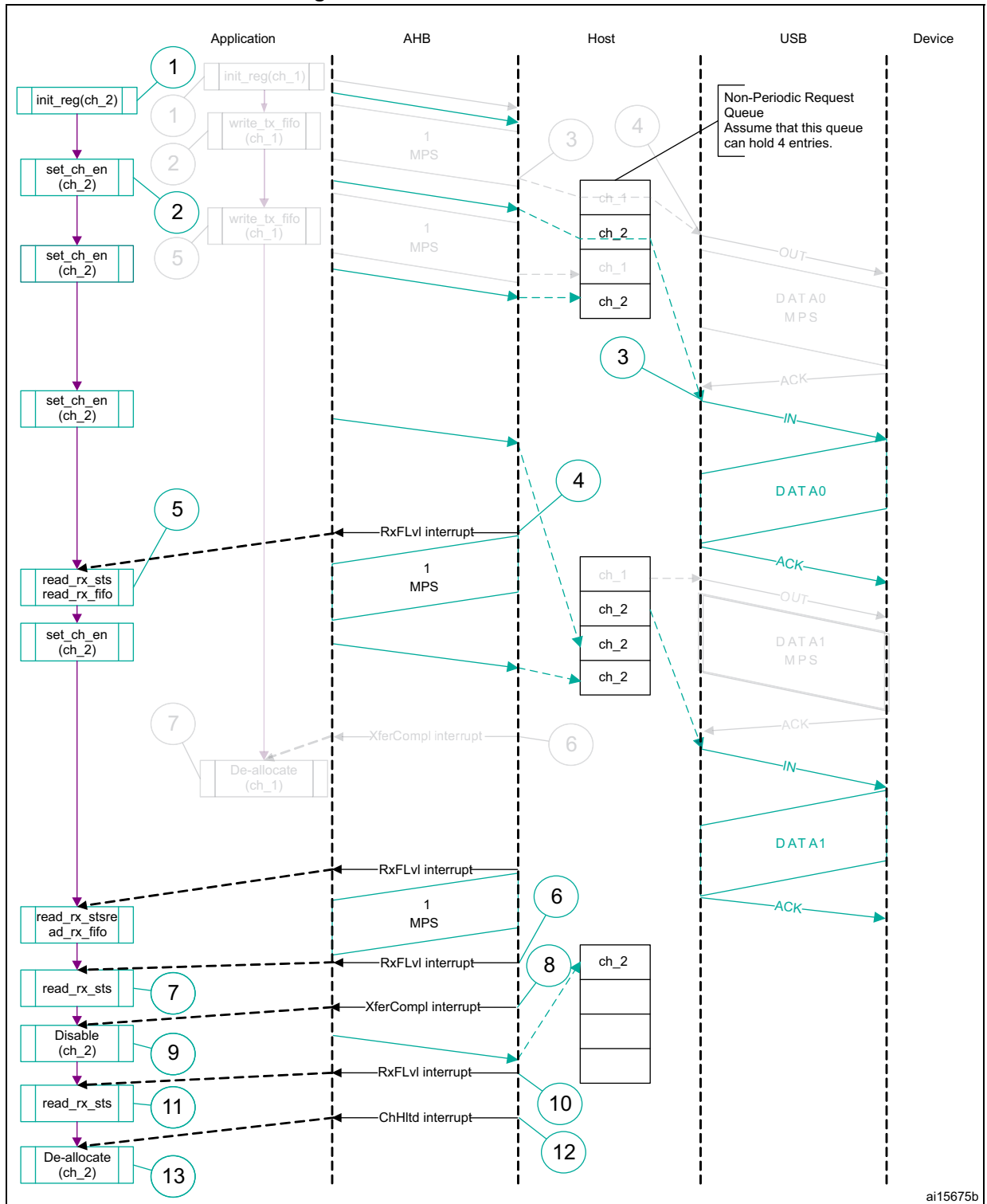
The application is expected to write the requests as and when the request queue space is available and until the XFRC interrupt is received.

- **Bulk and control IN transactions**

A typical bulk or control IN pipelined transaction-level operation is shown in [Figure 592](#). See channel 2 (ch\_2). The assumptions are:

- The application is attempting to receive two maximum-packet-size packets (transfer size = 1 024 bytes).
- The receive FIFO can contain at least one maximum-packet-size packet and two status words per packet (72 bytes for FS).
- The non-periodic request queue depth = 4.

Figure 592. Bulk/control IN transactions



1. The grayed elements are not relevant in the context of this figure.



The sequence of operations is as follows:

1. Initialize channel 2.
2. Set the CHENA bit in OTG\_HCCHAR2 to write an IN request to the non-periodic request queue.
3. The core attempts to send an IN token after completing the current OUT transaction.
4. The core generates an RXFLVL interrupt as soon as the received packet is written to the receive FIFO.
5. In response to the RXFLVL interrupt, mask the RXFLVL interrupt and read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. Following this, unmask the RXFLVL interrupt.
6. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO.
7. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in OTG\_GRXSTSR  $\neq$  0b0010).
8. The core generates the XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, disable the channel and stop writing the OTG\_HCCHAR2 register for further requests. The core writes a channel disable request to the non-periodic request queue as soon as the OTG\_HCCHAR2 register is written.
10. The core generates the RXFLVL interrupt as soon as the halt status is written to the receive FIFO.
11. Read and ignore the receive packet status.
12. The core generates a CHH interrupt as soon as the halt status is popped from the receive FIFO.
13. In response to the CHH interrupt, de-allocate the channel for other transfers.
14. Handling non-ACK responses

- **Control transactions**

Setup, data, and status stages of a control transfer must be performed as three separate transfers. setup-, data- or status-stage OUT transactions are performed similarly to the bulk OUT transactions explained previously. Data- or status-stage IN transactions are performed similarly to the bulk IN transactions explained previously. For all three stages, the application is expected to set the EPTYP field in OTG\_HCCHAR1 to control. During the setup stage, the application is expected to set the PID field in OTG\_HCTSIZ1 to SETUP.

- **Interrupt OUT transactions**

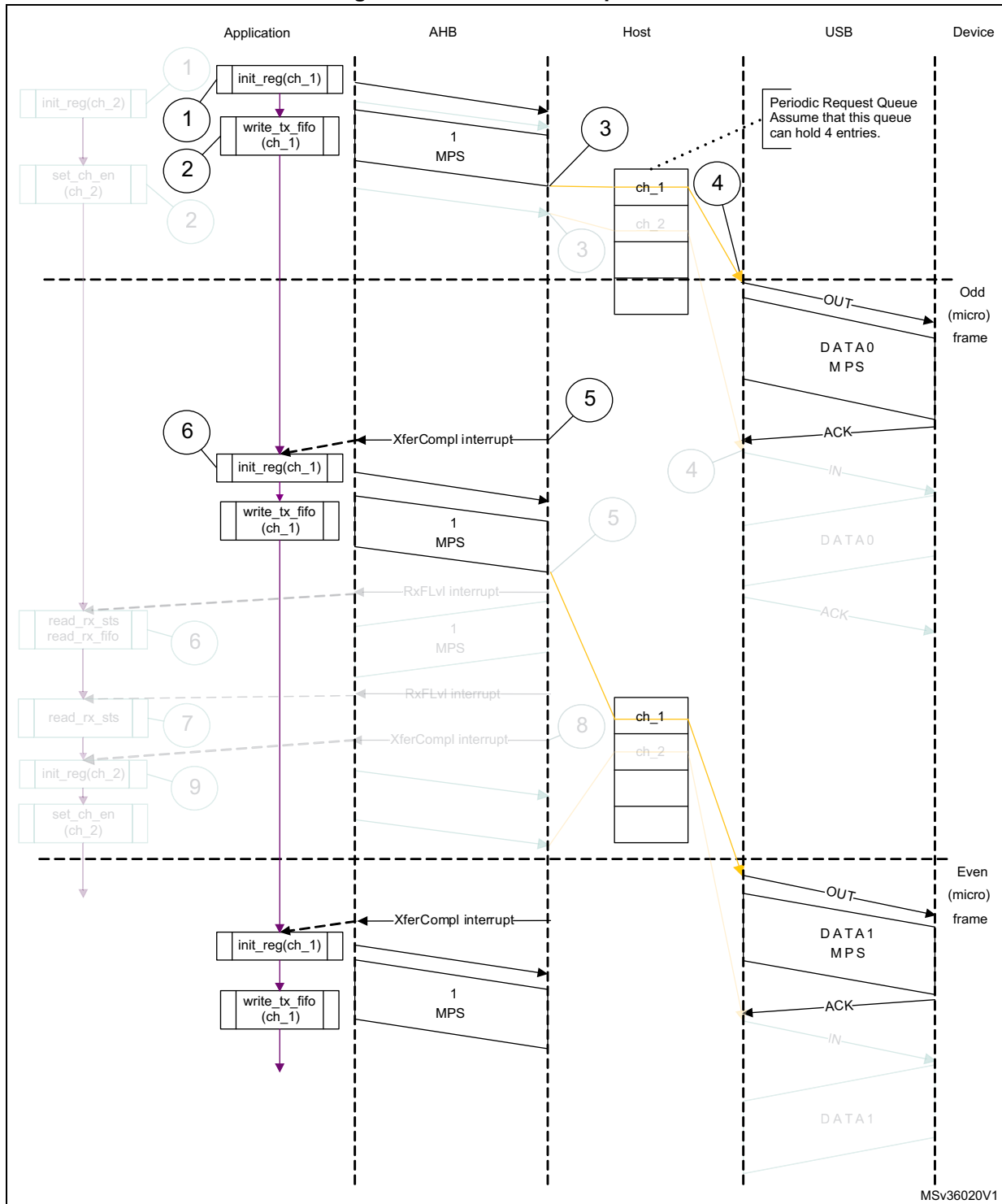
A typical interrupt OUT operation is shown in [Figure 593](#). The assumptions are:

- The application is attempting to send one packet in every frame (up to 1 maximum packet size), starting with the odd frame (transfer size = 1 024 bytes)
- The periodic transmit FIFO can hold one packet (1 KB)
- Periodic request queue depth = 4

The sequence of operations is as follows:

1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG\_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG\_FS host writes an entry to the periodic request queue.
4. The OTG\_FS host attempts to send an OUT token in the next (odd) frame.
5. The OTG\_FS host generates an XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.

Figure 593. Normal interrupt OUT



MSv36020V1

1. The grayed elements are not relevant in the context of this figure.

- **Interrupt service routine for interrupt OUT/IN transactions**

- a) **Interrupt OUT**

Unmask (NAK/TXERR/STALL/XFRC/FRMOR)

```
if (XFRC)
{
Reset Error Count
Mask ACK
De-allocate Channel
}
else
if (STALL or FRMOR)
{
Mask ACK
Unmask CHH
Disable Channel
if (STALL)
{
Transfer Done = 1
}
}
else
if (NAK or TXERR)
{
Rewind Buffer Pointers
Reset Error Count
Mask ACK
Unmask CHH
Disable Channel
}
else
if (CHH)
{
Mask CHH
if (Transfer Done or (Error_count == 3))
{
De-allocate Channel
}
else
{
Re-initialize Channel (in next b_interval - 1 Frame)
}
}
else
if (ACK)
{
Reset Error Count
Mask ACK
}
```

The application uses the NPTXFE interrupt in OTG\_GINTSTS to find the transmit FIFO space.

Interrupt IN

Unmask (NAK/TXERR/XFRC/BBERR/STALL/FRMOR/DTERR)

```

if (XFRC)
{
  Reset Error Count
  Mask ACK
  if (OTG_HCTSIZx.PKTCNT == 0)
  {
    De-allocate Channel
  }
else
  {
    Transfer Done = 1
    Unmask CHH
    Disable Channel
  }
}
else
  if (STALL or FRMOR or NAK or DTERR or BBERR)
  {
    Mask ACK
    Unmask CHH
    Disable Channel
    if (STALL or BBERR)
    {
      Reset Error Count
      Transfer Done = 1
    }
    else
      if (!FRMOR)
      {
        Reset Error Count
      }
  }
else
  if (TXERR)
  {
    Increment Error Count
    Unmask ACK
    Unmask CHH
    Disable Channel
  }
else

```

```

if (CHH)
{
Mask CHH
if (Transfer Done or (Error_count == 3))
{
De-allocate Channel
}
else
Re-initialize Channel (in next b_interval - 1 /Frame)
}
}
else
if (ACK)
{
Reset Error Count
Mask ACK
}

```

- **Interrupt IN transactions**

The assumptions are:

- The application is attempting to receive one packet (up to 1 maximum packet size) in every frame, starting with odd (transfer size = 1 024 bytes).
- The receive FIFO can hold at least one maximum-packet-size packet and two status words per packet (1 031 bytes).
- Periodic request queue depth = 4.

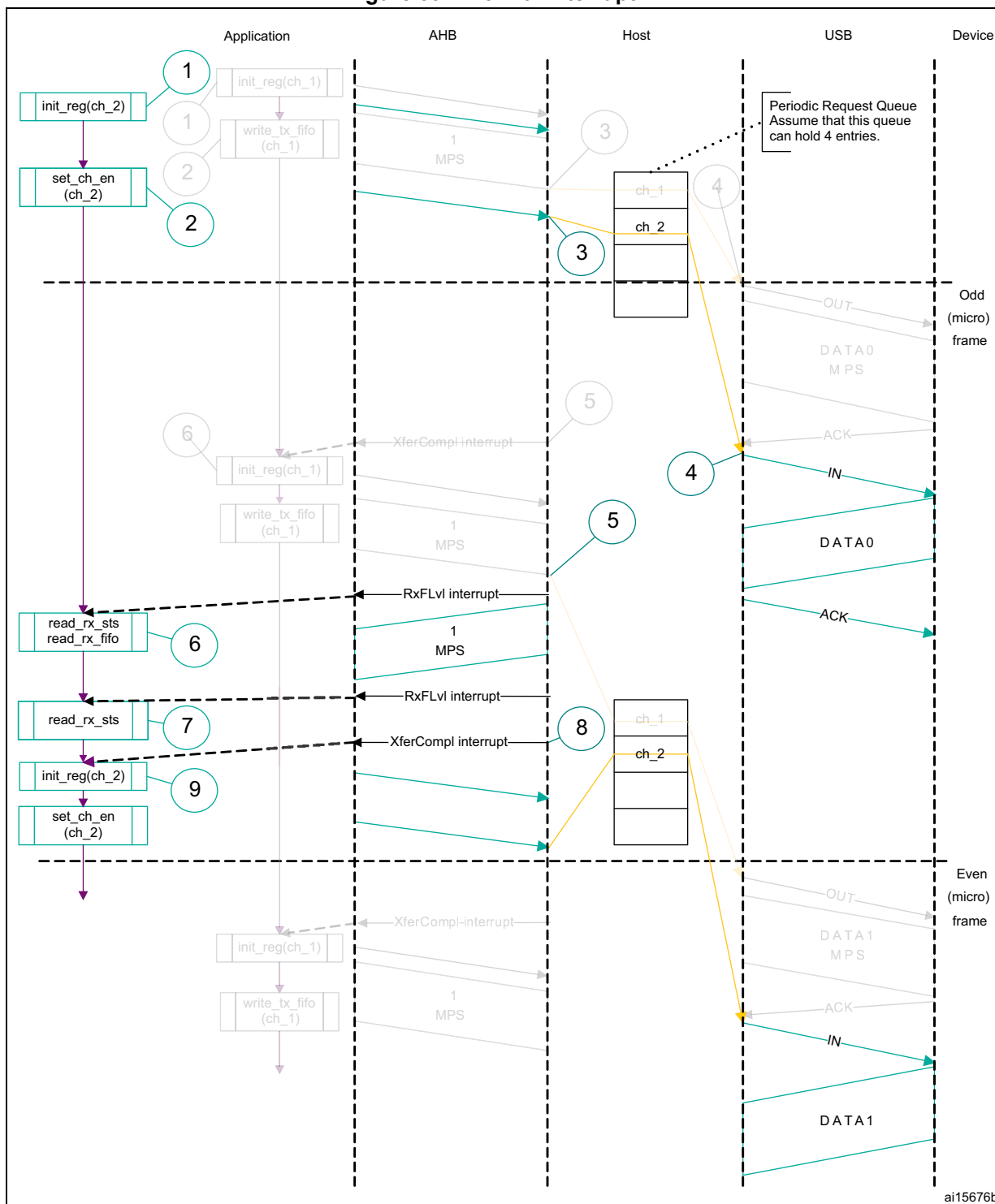
- **Normal interrupt IN operation**

The sequence of operations is as follows:

1. Initialize channel 2. The application must set the ODDFRM bit in OTG\_HCCHAR2.
2. Set the CHENA bit in OTG\_HCCHAR2 to write an IN request to the periodic request queue.
3. The OTG\_FS host writes an IN request to the periodic request queue for each OTG\_HCCHAR2 register write with the CHENA bit set.
4. The OTG\_FS host attempts to send an IN token in the next (odd) frame.
5. As soon as the IN packet is received and written to the receive FIFO, the OTG\_FS host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask after reading the entire packet.
7. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in GRXSTSR ≠ 0b0010).
8. The core generates an XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, read the PKTCNT field in OTG\_HCTSIZ2. If the PKTCNT bit in OTG\_HCTSIZ2 is not equal to 0, disable the channel before re-

initializing the channel for the next transfer, if any). If PKTCNT bit in OTG\_HCTSIZ2 = 0, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG\_HCCHAR2.

Figure 594. Normal interrupt IN



1. The grayed elements are not relevant in the context of this figure.

- **Isochronous OUT transactions**

A typical isochronous OUT operation is shown in [Figure 594](#). The assumptions are:

- The application is attempting to send one packet every frame (up to 1 maximum



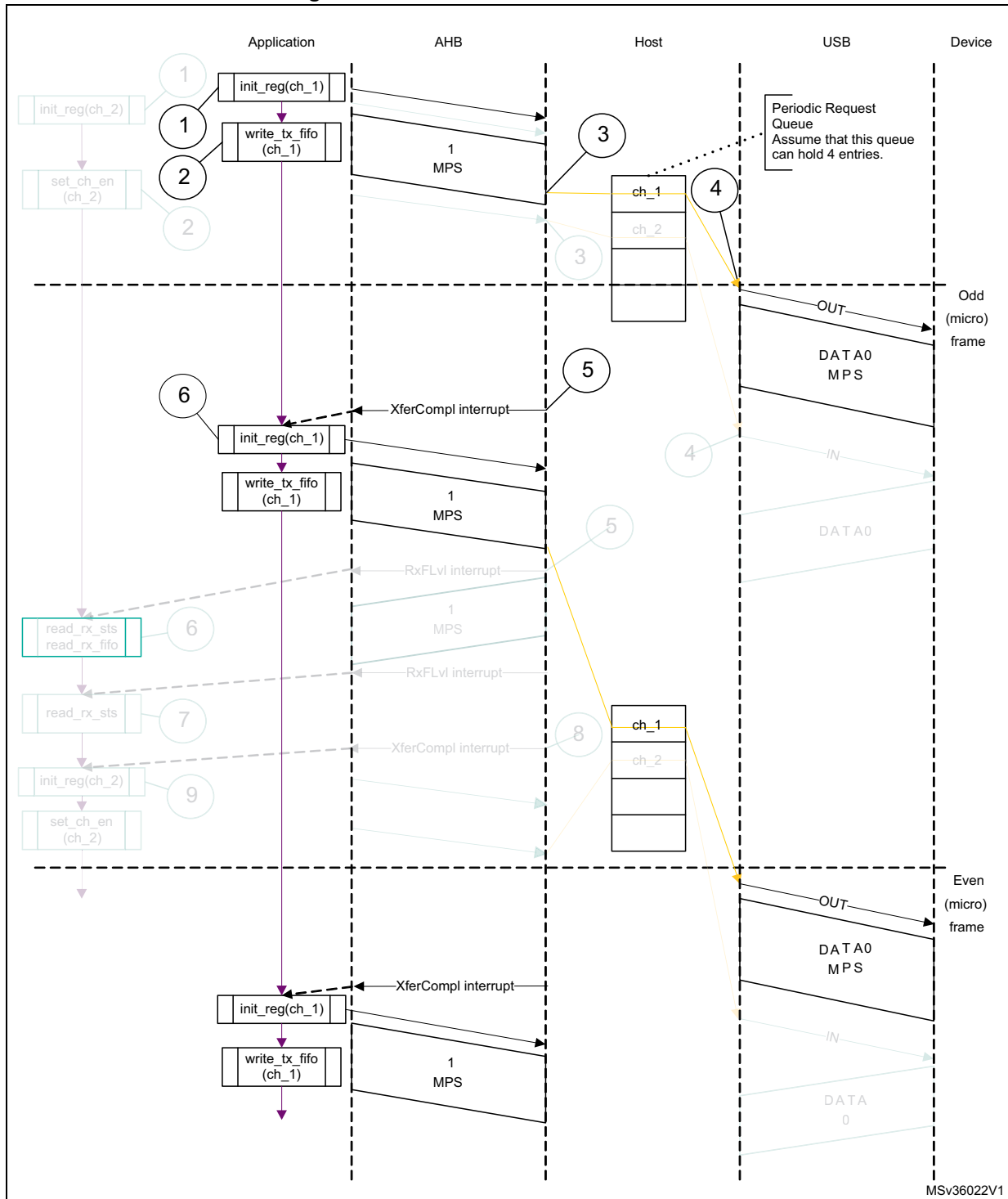
packet size), starting with an odd frame. (transfer size = 1 024 bytes).

- The periodic transmit FIFO can hold one packet (1 KB).
- Periodic request queue depth = 4.

The sequence of operations is as follows:

1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG\_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG\_FS host writes an entry to the periodic request queue.
4. The OTG\_FS host attempts to send the OUT token in the next frame (odd).
5. The OTG\_FS host generates the XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.
7. Handling non-ACK responses

Figure 595. Isochronous OUT transactions



MSv36022V1

1. The grayed elements are not relevant in the context of this figure.

- **Interrupt service routine for isochronous OUT/IN transactions**

Code sample: isochronous OUT

```

Unmask (FRMOR/XFRC)
if (XFRC)
    
```

```
    {
        De-allocate Channel
    }
else
    if (FRMOR)
    {
        Unmask CHH
        Disable Channel
    }
else
    if (CHH)
    {
        Mask CHH
        De-allocate Channel
    }
Code sample: Isochronous IN
Unmask (TXERR/XFRC/FRMOR/BBERR)
if (XFRC or FRMOR)
{
    if (XFRC and (OTG_HCTSIZx.PKTCNT == 0))
    {
        Reset Error Count
        De-allocate Channel
    }
else
    {
        Unmask CHH
        Disable Channel
    }
}
else
    if (TXERR or BBERR)
    {
        Increment Error Count
        Unmask CHH
        Disable Channel
    }
else
    if (CHH)
    {
        Mask CHH
        if (Transfer Done or (Error_count == 3))
        {
            De-allocate Channel
        }
    }
}
```

```

else
{
    Re-initialize Channel
}
}

```

- **Isochronous IN transactions**

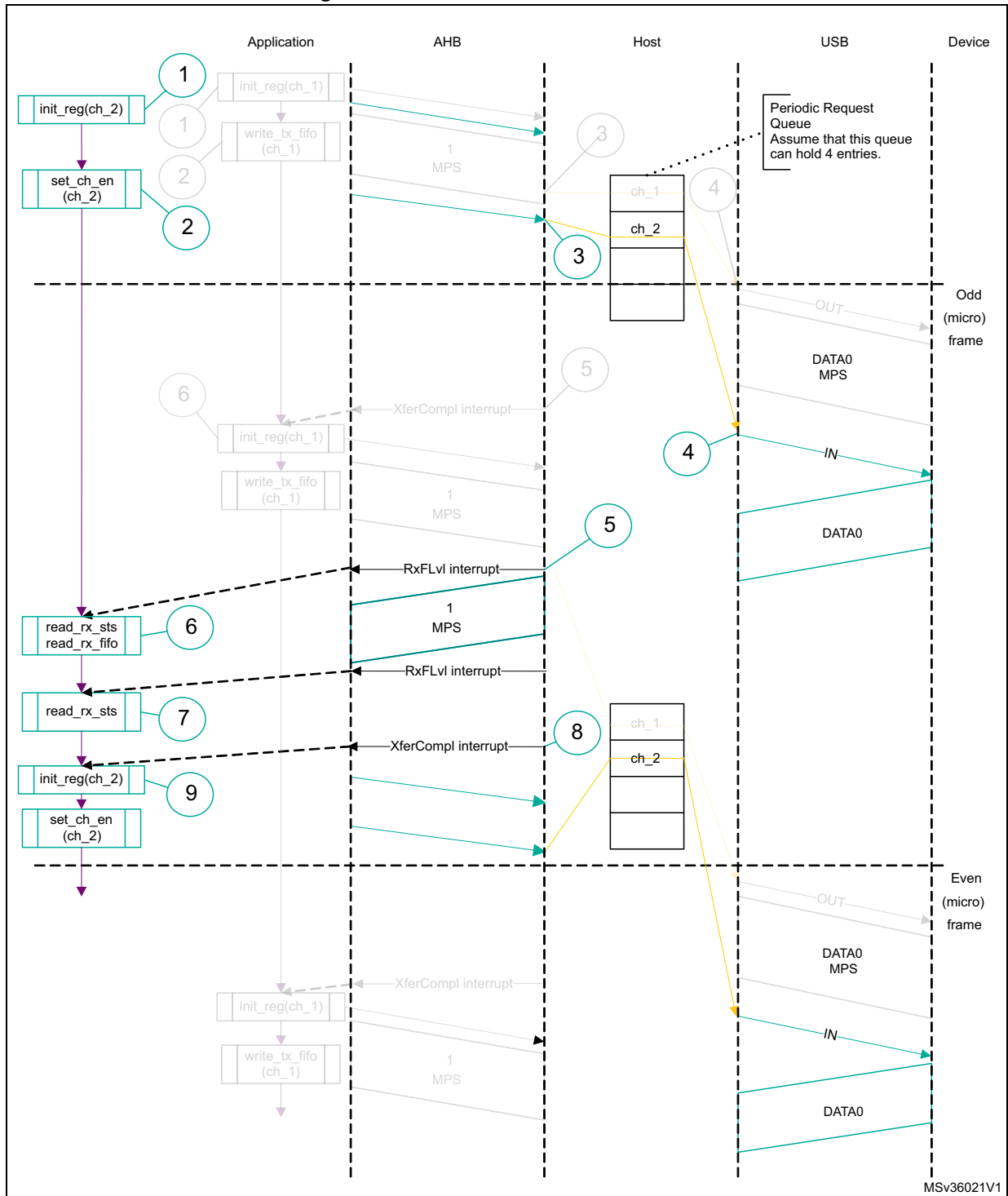
The assumptions are:

- The application is attempting to receive one packet (up to 1 maximum packet size) in every frame starting with the next odd frame (transfer size = 1 024 bytes).
- The receive FIFO can hold at least one maximum-packet-size packet and two status word per packet (1 031 bytes).
- Periodic request queue depth = 4.

The sequence of operations is as follows:

1. Initialize channel 2. The application must set the ODDFRM bit in OTG\_HCCHAR2.
2. Set the CHENA bit in OTG\_HCCHAR2 to write an IN request to the periodic request queue.
3. The OTG\_FS host writes an IN request to the periodic request queue for each OTG\_HCCHAR2 register write with the CHENA bit set.
4. The OTG\_FS host attempts to send an IN token in the next odd frame.
5. As soon as the IN packet is received and written to the receive FIFO, the OTG\_FS host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask it after reading the entire packet.
7. The core generates an RXFLVL interrupt for the transfer completion status entry in the receive FIFO. This time, the application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS bit in OTG\_GRXSTSR ≠ 0b0010).
8. The core generates an XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, read the PKTCNT field in OTG\_HCTSIZ2. If PKTCNT ≠ 0 in OTG\_HCTSIZ2, disable the channel before re-initializing the channel for the next transfer, if any. If PKTCNT = 0 in OTG\_HCTSIZ2, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG\_HCCHAR2.

Figure 596. Isochronous IN transactions



1. The grayed elements are not relevant in the context of this figure.

- Selecting the queue depth**

Choose the periodic and non-periodic request queue depths carefully to match the number of periodic/non-periodic endpoints accessed.

The non-periodic request queue depth affects the performance of non-periodic

transfers. The deeper the queue (along with sufficient FIFO size), the more often the core is able to pipeline non-periodic transfers. If the queue size is small, the core is able to put in new requests only when the queue space is freed up.

The core's periodic request queue depth is critical to perform periodic transfers as scheduled. Select the periodic queue depth, based on the number of periodic transfers scheduled in a microframe. If the periodic request queue depth is smaller than the periodic transfers scheduled in a microframe, a frame overrun condition occurs.

- **Handling babble conditions**

OTG\_FS controller handles two cases of babble: packet babble and port babble.

Packet babble occurs if the device sends more data than the maximum packet size for the channel. Port babble occurs if the core continues to receive data from the device at EOF2 (the end of frame 2, which is very close to SOF).

When OTG\_FS controller detects a packet babble, it stops writing data into the Rx buffer and waits for the end of packet (EOP). When it detects an EOP, it flushes already written data in the Rx buffer and generates a Babble interrupt to the application.

When OTG\_FS controller detects a port babble, it flushes the Rx FIFO and disables the port. The core then generates a port disabled interrupt (HPRTINT in OTG\_GINTSTS, PENCHNG in OTG\_HPRT). On receiving this interrupt, the application must determine that this is not due to an overcurrent condition (another cause of the port disabled interrupt) by checking POCA in OTG\_HPRT, then perform a soft reset. The core does not send any more tokens after it has detected a port babble condition.

## 50.16.5 Device programming model

### Endpoint initialization on USB reset

1. Set the NAK bit for all OUT endpoints
  - SNAK = 1 in OTG\_DOEPCtx (for all OUT endpoints)
2. Unmask the following interrupt bits
  - INEP0 = 1 in OTG\_DAINMSK (control 0 IN endpoint)
  - OUTEP0 = 1 in OTG\_DAINMSK (control 0 OUT endpoint)
  - STUPM = 1 in OTG\_DOEPMASK
  - XFRCM = 1 in OTG\_DOEPMASK
  - XFRCM = 1 in OTG\_DIEPMASK
  - TOM = 1 in OTG\_DIEPMASK
3. Set up the data FIFO RAM for each of the FIFOs
  - Program the OTG\_GRXFSIZ register, to be able to receive control OUT data and setup data. If thresholding is not enabled, at a minimum, this must be equal to 1 max packet size of control endpoint 0 + 2 words (for the status of the control OUT data packet) + 10 words (for setup packets).
  - Program the OTG\_DIEPTXF0 register (depending on the FIFO number chosen) to be able to transmit control IN data. At a minimum, this must be equal to 1 max packet size of control endpoint 0.
4. Program the following fields in the endpoint-specific registers for control OUT endpoint 0 to receive a SETUP packet
  - STUPCNT = 3 in OTG\_DOEPTSIZ0 (to receive up to 3 back-to-back SETUP packets)

At this point, all initialization required to receive SETUP packets is done.

### Endpoint initialization on enumeration completion

1. On the Enumeration Done interrupt (ENUMDNE in OTG\_GINTSTS), read the OTG\_DSTS register to determine the enumeration speed.
2. Program the MPSIZ field in OTG\_DIEPCTL0 to set the maximum packet size. This step configures control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed.

At this point, the device is ready to receive SOF packets and is configured to perform control transfers on control endpoint 0.

### Endpoint initialization on SetAddress command

This section describes what the application must do when it receives a SetAddress command in a SETUP packet.

1. Program the OTG\_DCFG register with the device address received in the SetAddress command
2. Program the core to send out a status IN packet

### Endpoint initialization on SetConfiguration/SetInterface command

This section describes what the application must do when it receives a SetConfiguration or SetInterface command in a SETUP packet.

1. When a SetConfiguration command is received, the application must program the endpoint registers to configure them with the characteristics of the valid endpoints in the new configuration.
2. When a SetInterface command is received, the application must program the endpoint registers of the endpoints affected by this command.
3. Some endpoints that were active in the prior configuration or alternate setting are not valid in the new configuration or alternate setting. These invalid endpoints must be deactivated.
4. Unmask the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the OTG\_DAINMSK register.
5. Set up the data FIFO RAM for each FIFO.
6. After all required endpoints are configured; the application must program the core to send a status IN packet.

At this point, the device core is configured to receive and transmit any type of data packet.

### Endpoint activation

This section describes the steps required to activate a device endpoint or to configure an existing device endpoint to a new type.

1. Program the characteristics of the required endpoint into the following fields of the OTG\_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG\_DOEPCTLx register (for OUT or bidirectional endpoints).
  - Maximum packet size
  - USB active endpoint = 1
  - Endpoint start data toggle (for interrupt and bulk endpoints)
  - Endpoint type
  - Tx FIFO number
2. Once the endpoint is activated, the core starts decoding the tokens addressed to that endpoint and sends out a valid handshake for each valid token received for the endpoint.

### Endpoint deactivation

This section describes the steps required to deactivate an existing endpoint.

1. In the endpoint to be deactivated, clear the USB active endpoint bit in the OTG\_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG\_DOEPCTLx register (for OUT or bidirectional endpoints).
2. Once the endpoint is deactivated, the core ignores tokens addressed to that endpoint, which results in a timeout on the USB.

*Note:* The application must meet the following conditions to set up the device core to handle traffic:

*NPTXFEM and RXFLVLM in the OTG\_GINTMSK register must be cleared.*

### Operational model

SETUP and OUT data transfers:

This section describes the internal data flow and application-level operations during data OUT transfers and SETUP transactions.

- **Packet read**

This section describes how to read packets (OUT data and SETUP packets) from the receive FIFO.

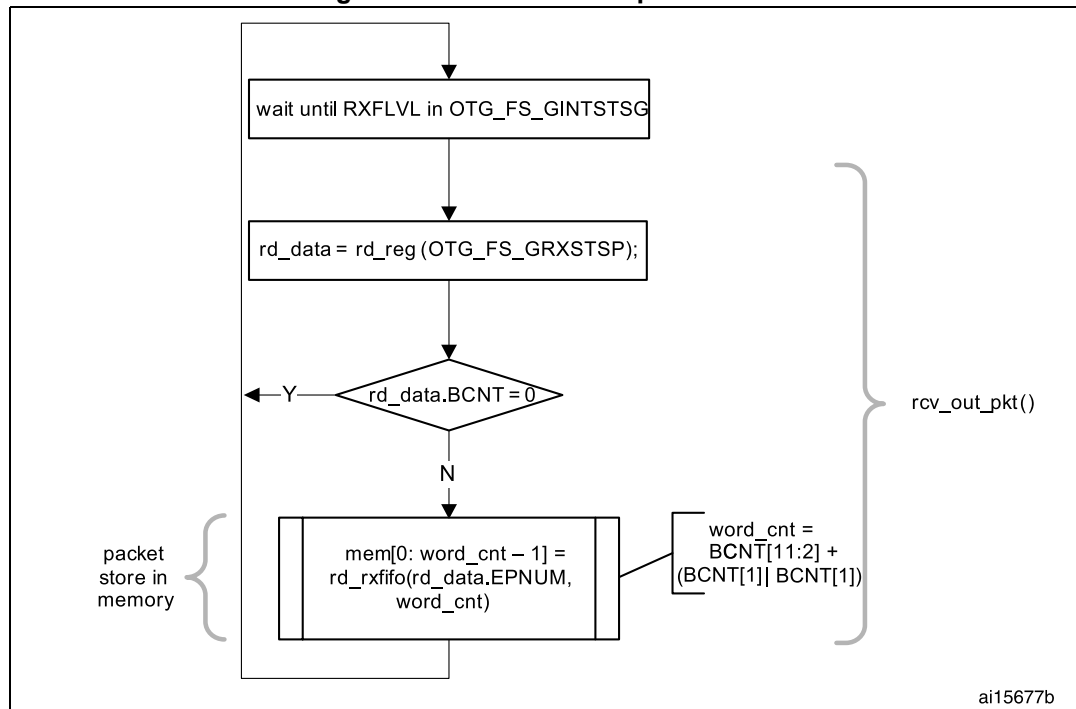
1. On catching an RXFLVL interrupt (OTG\_GINTSTS register), the application must read the receive status pop register (OTG\_GRXSTSP).
2. The application can mask the RXFLVL interrupt (in OTG\_GINTSTS) by writing to RXFLVLM = 0 (in OTG\_GINTMSK), until it has read the packet from the receive FIFO.
3. If the received packet's byte count is not 0, the byte count amount of data is popped from the receive data FIFO and stored in memory. If the received packet byte count is 0, no data is popped from the receive data FIFO.
4. The receive status readout of the packet of FIFO indicates one of the following:
  - a) Global OUT NAK pattern:  
PKTSTS = Global OUT NAK, BCNT = 0x000, EPNUM = (0x0),  
DPID = (0b00).  
These data indicate that the global OUT NAK bit has taken effect.
  - b) SETUP packet pattern:  
PKTSTS = SETUP, BCNT = 0x008, EPNUM = Control EP Num,



- DPID = DATA0. These data indicate that a SETUP packet for the specified endpoint is now available for reading from the receive FIFO.
- c) Setup stage done pattern:  
 PKTSTS = Setup Stage Done, BCNT = 0x0, EPNUM = Control EP Num, DPID = (0b00).  
 These data indicate that the setup stage for the specified endpoint has completed and the data stage has started. After this entry is popped from the receive FIFO, the core asserts a setup interrupt on the specified control OUT endpoint.
  - d) Data OUT packet pattern:  
 PKTSTS = DataOUT, BCNT = size of the received data OUT packet ( $0 \leq BCNT \leq 1024$ ), EPNUM = EPNUM on which the packet was received, DPID = Actual Data PID.
  - e) Data transfer completed pattern:  
 PKTSTS = Data OUT transfer done, BCNT = 0x0, EPNUM = OUT EP Num on which the data transfer is complete, DPID = (0b00).  
 These data indicate that an OUT data transfer for the specified OUT endpoint has completed. After this entry is popped from the receive FIFO, the core asserts a transfer completed interrupt on the specified OUT endpoint.
5. After the data payload is popped from the receive FIFO, the RXFLVL interrupt (OTG\_GINTSTS) must be unmasked.
  6. Steps 1–5 are repeated every time the application detects assertion of the interrupt line due to RXFLVL in OTG\_GINTSTS. Reading an empty receive FIFO can result in undefined core behavior.

Figure 597 provides a flowchart of the above procedure.

Figure 597. Receive FIFO packet read



**SETUP transactions**

This section describes how the core handles SETUP packets and the application's sequence for handling SETUP transactions.

- **Application requirements**

1. To receive a SETUP packet, the STUPCNT field (OTG\_DOEPTSIZE<sub>x</sub>) in a control OUT endpoint must be programmed to a non-zero value. When the application programs the STUPCNT field to a non-zero value, the core receives SETUP packets and writes them to the receive FIFO, irrespective of the NAK status and EPENA bit setting in OTG\_DOEPCTL<sub>x</sub>. The STUPCNT field is decremented every time the control endpoint receives a SETUP packet. If the STUPCNT field is not programmed to a proper value before receiving a SETUP packet, the core still receives the SETUP packet and decrements the STUPCNT field, but the application may not be able to determine the correct number of SETUP packets received in the setup stage of a control transfer.
  - STUPCNT = 3 in OTG\_DOEPTSIZE<sub>x</sub>
2. The application must always allocate some extra space in the receive data FIFO, to be able to receive up to three SETUP packets on a control endpoint.
  - The space to be reserved is 10 words. Three words are required for the first SETUP packet, 1 word is required for the setup stage done word and 6 words are required to store two extra SETUP packets among all control endpoints.
  - 3 words per SETUP packet are required to store 8 bytes of SETUP data and 4 bytes of SETUP status (setup packet pattern). The core reserves this space in the receive data FIFO to write SETUP data only, and never uses this space for data packets.
3. The application must read the 2 words of the SETUP packet from the receive FIFO.
4. The application must read and discard the setup stage done word from the receive FIFO.

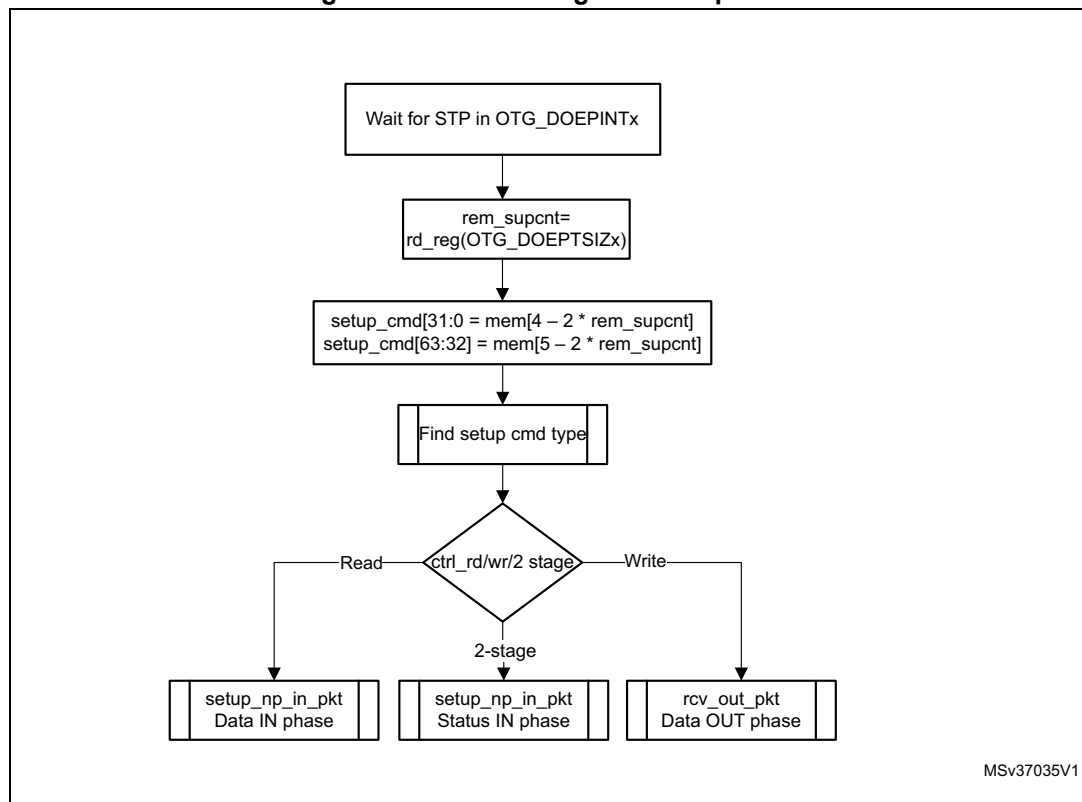
- **Internal data flow**

1. When a SETUP packet is received, the core writes the received data to the receive FIFO, without checking for available space in the receive FIFO and irrespective of the endpoint's NAK and STALL bit settings.
  - The core internally sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.
2. For every SETUP packet received on the USB, 3 words of data are written to the receive FIFO, and the STUPCNT field is decremented by 1.
  - The first word contains control information used internally by the core
  - The second word contains the first 4 bytes of the SETUP command
  - The third word contains the last 4 bytes of the SETUP command
3. When the setup stage changes to a data IN/OUT stage, the core writes an entry (setup stage done word) to the receive FIFO, indicating the completion of the setup stage.
4. On the AHB side, SETUP packets are emptied by the application.
5. When the application pops the setup stage done word from the receive FIFO, the core interrupts the application with an STUP interrupt (OTG\_DOEPINT<sub>x</sub>), indicating it can process the received SETUP packet.
6. The core clears the endpoint enable bit for control OUT endpoints.

- **Application programming sequence**

1. Program the OTG\_DOEPTSIZx register.
  - STUPCNT = 3
2. Wait for the RXFLVL interrupt (OTG\_GINTSTS) and empty the data packets from the receive FIFO.
3. Assertion of the STUP interrupt (OTG\_DOEPINTx) marks a successful completion of the SETUP data transfer.
  - On this interrupt, the application must read the OTG\_DOEPTSIZx register to determine the number of SETUP packets received and process the last received SETUP packet.

**Figure 598. Processing a SETUP packet**



**• Handling more than three back-to-back SETUP packets**

Per the USB 2.0 specification, normally, during a SETUP packet error, a host does not send more than three back-to-back SETUP packets to the same endpoint. However, the USB 2.0 specification does not limit the number of back-to-back SETUP packets a host can send to the same endpoint. When this condition occurs, the OTG\_FS controller generates an interrupt (B2BSTUP in OTG\_DOEPINTx).

**• Setting the global OUT NAK**

Internal data flow:

1. When the application sets the Global OUT NAK (SGONAK bit in OTG\_DCTL), the core stops writing data, except SETUP packets, to the receive FIFO. Irrespective of the

space availability in the receive FIFO, non-isochronous OUT tokens receive a NAK handshake response, and the core ignores isochronous OUT data packets

2. The core writes the Global OUT NAK pattern to the receive FIFO. The application must reserve enough receive FIFO space to write this data pattern.
3. When the application pops the Global OUT NAK pattern word from the receive FIFO, the core sets the GONAKEFF interrupt (OTG\_GINTSTS).
4. Once the application detects this interrupt, it can assume that the core is in Global OUT NAK mode. The application can clear this interrupt by clearing the SGONAK bit in OTG\_DCTL.

Application programming sequence:

1. To stop receiving any kind of data in the receive FIFO, the application must set the Global OUT NAK bit by programming the following field:
  - SGONAK = 1 in OTG\_DCTL
2. Wait for the assertion of the GONAKEFF interrupt in OTG\_GINTSTS. When asserted, this interrupt indicates that the core has stopped receiving any type of data except SETUP packets.
3. The application can receive valid OUT packets after it has set SGONAK in OTG\_DCTL and before the core asserts the GONAKEFF interrupt (OTG\_GINTSTS).
4. The application can temporarily mask this interrupt by writing to the GONAKEFFM bit in the OTG\_GINTMSK register.
  - GONAKEFFM = 0 in the OTG\_GINTMSK register
5. Whenever the application is ready to exit the Global OUT NAK mode, it must clear the SGONAK bit in OTG\_DCTL. This also clears the GONAKEFF interrupt (OTG\_GINTSTS).
  - CGONAK = 1 in OTG\_DCTL
6. If the application has masked this interrupt earlier, it must be unmasked as follows:
  - GONAKEFFM = 1 in OTG\_GINTMSK

- **Disabling an OUT endpoint**

The application must use this sequence to disable an OUT endpoint that it has enabled.

Application programming sequence:

1. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core.
  - SGONAK = 1 in OTG\_DCTL
2. Wait for the GONAKEFF interrupt (OTG\_GINTSTS)
3. Disable the required OUT endpoint by programming the following fields:
  - EPDIS = 1 in OTG\_DOEPCTLx
  - SNAK = 1 in OTG\_DOEPCTLx
4. Wait for the EPDISD interrupt (OTG\_DOEPINTx), which indicates that the OUT endpoint is completely disabled. When the EPDISD interrupt is asserted, the core also clears the following bits:
  - EPDIS = 0 in OTG\_DOEPCTLx
  - EPENA = 0 in OTG\_DOEPCTLx
5. The application must clear the Global OUT NAK bit to start receiving data from other non-disabled OUT endpoints.
  - SGONAK = 0 in OTG\_DCTL

- **Generic non-isochronous OUT data transfers**

This section describes a regular non-isochronous OUT data transfer (control, bulk, or interrupt).

Application requirements:

1. Before setting up an OUT transfer, the application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer.
2. For OUT transfers, the transfer size field in the endpoint's transfer size register must be a multiple of the maximum packet size of the endpoint, adjusted to the word boundary.
  - $\text{transfer size}[\text{EPNUM}] = n \times (\text{MPSIZ}[\text{EPNUM}] + 4 - (\text{MPSIZ}[\text{EPNUM}] \bmod 4))$
  - $\text{packet count}[\text{EPNUM}] = n$
  - $n > 0$
3. On any OUT endpoint interrupt, the application must read the endpoint's transfer size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
  - $\text{Payload size in memory} = \text{application programmed initial transfer size} - \text{core updated final transfer size}$
  - $\text{Number of USB packets in which this payload was received} = \text{application programmed initial packet count} - \text{core updated final packet count}$

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet

- (maximum packet size or short packet) written to the receive FIFO decrements the packet count field for that endpoint by 1.
- OUT data packets received with bad data CRC are flushed from the receive FIFO automatically.
  - After sending an ACK for the packet on the USB, the core discards non-isochronous OUT data packets that the host, which cannot detect the ACK, re-sends. The application does not detect multiple back-to-back data OUT packets on the same endpoint with the same data PID. In this case the packet count is not decremented.
  - If there is no space in the receive FIFO, isochronous or non-isochronous data packets are ignored and not written to the receive FIFO. Additionally, non-isochronous OUT tokens receive a NAK handshake reply.
  - In all the above three cases, the packet count is not decremented because no data are written to the receive FIFO.
3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or non-isochronous data packets are ignored and not written to the receive FIFO, and non-isochronous OUT tokens receive a NAK handshake reply.
  4. After the data are written to the receive FIFO, the application reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
  5. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.
  6. The OUT data transfer completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions:
    - The transfer size is 0 and the packet count is 0
    - The last OUT data packet written to the receive FIFO is a short packet ( $0 \leq \text{packet size} < \text{maximum packet size}$ )
  7. When either the application pops this entry (OUT data transfer completed), a transfer completed interrupt is generated for the endpoint and the endpoint enable is cleared.

#### Application programming sequence:

1. Program the OTG\_DOEPTSIZE register for the transfer size and the corresponding packet count.
2. Program the OTG\_DOEPCTL register with the endpoint characteristics, and set the EPENA and CNAK bits.
  - EPENA = 1 in OTG\_DOEPCTL
  - CNAK = 1 in OTG\_DOEPCTL
3. Wait for the RXFLVL interrupt (in OTG\_GINTSTS) and empty the data packets from the receive FIFO.
  - This step can be repeated many times, depending on the transfer size.
4. Asserting the XFRC interrupt (OTG\_DOEPINT) marks a successful completion of the non-isochronous OUT data transfer.
5. Read the OTG\_DOEPTSIZE register to determine the size of the received data payload.

- **Generic isochronous OUT data transfer**

This section describes a regular isochronous OUT data transfer.

## Application requirements:

1. All the application requirements for non-isochronous OUT data transfers also apply to isochronous OUT data transfers.
2. For isochronous OUT data transfers, the transfer size and packet count fields must always be set to the number of maximum-packet-size packets that can be received in a single frame and no more. Isochronous OUT data transfers cannot span more than 1 frame.
3. The application must read all isochronous OUT data packets from the receive FIFO (data and status) before the end of the periodic frame (EOPF interrupt in OTG\_GINTSTS).
4. To receive data in the following frame, an isochronous OUT endpoint must be enabled after the EOPF (OTG\_GINTSTS) and before the SOF (OTG\_GINTSTS).

## Internal data flow:

1. The internal data flow for isochronous OUT endpoints is the same as that for non-isochronous OUT endpoints, but for a few differences.
2. When an isochronous OUT endpoint is enabled by setting the endpoint enable and clearing the NAK bits, the Even/Odd frame bit must also be set appropriately. The core receives data on an isochronous OUT endpoint in a particular frame only if the following condition is met:
  - $EONUM$  (in OTG\_DOEPCTLx) =  $FNSOF[0]$  (in OTG\_DSTS)
3. When the application completely reads an isochronous OUT data packet (data and status) from the receive FIFO, the core updates the RXDPID field in OTG\_DOEPTSIZx with the data PID of the last isochronous OUT data packet read from the receive FIFO.

## Application programming sequence:

1. Program the OTG\_DOEPTSIZE register for the transfer size and the corresponding packet count
2. Program the OTG\_DOEPCTL register with the endpoint characteristics and set the endpoint enable, ClearNAK, and Even/Odd frame bits.
  - EPENA = 1
  - CNAK = 1
  - EONUM = (0: Even/1: Odd)
3. Wait for the RXFLVL interrupt (in OTG\_GINTSTS) and empty the data packets from the receive FIFO
  - This step can be repeated many times, depending on the transfer size.
4. The assertion of the XFRC interrupt (in OTG\_DOEPINT) marks the completion of the isochronous OUT data transfer. This interrupt does not necessarily mean that the data in memory are good.
5. This interrupt cannot always be detected for isochronous OUT transfers. Instead, the application can detect the INCOMPISOOUT interrupt in OTG\_GINTSTS.
6. Read the OTG\_DOEPTSIZE register to determine the size of the received transfer and to determine the validity of the data received in the frame. The application must treat the data received in memory as valid only if one of the following conditions is met:
  - RXDPID = DATA0 (in OTG\_DOEPTSIZE) and the number of USB packets in which this payload was received = 1
  - RXDPID = DATA1 (in OTG\_DOEPTSIZE) and the number of USB packets in which this payload was received = 2The number of USB packets in which this payload was received =  
Application programmed initial packet count – core updated final packet count  
The application can discard invalid data packets.

- **Incomplete isochronous OUT data transfers**

This section describes the application programming sequence when isochronous OUT data packets are dropped inside the core.

Internal data flow:

1. For isochronous OUT endpoints, the XFRC interrupt (in OTG\_DOEPINT) may not always be asserted. If the core drops isochronous OUT data packets, the application could fail to detect the XFRC interrupt (OTG\_DOEPINT) under the following circumstances:
  - When the receive FIFO cannot accommodate the complete ISO OUT data packet, the core drops the received ISO OUT data
  - When the isochronous OUT data packet is received with CRC errors
  - When the isochronous OUT token received by the core is corrupted
  - When the application is very slow in reading the data from the receive FIFO
2. When the core detects an end of periodic frame before transfer completion to all isochronous OUT endpoints, it asserts the incomplete isochronous OUT data interrupt (INCOMPISOOUT in OTG\_GINTSTS), indicating that an XFRC interrupt (in OTG\_DOEPINT) is not asserted on at least one of the isochronous OUT endpoints. At this point, the endpoint with the incomplete transfer remains enabled, but no active transfers remain in progress on this endpoint on the USB.



Application programming sequence:

1. Asserting the INCOMPISOOUT interrupt (OTG\_GINTSTS) indicates that in the current frame, at least one isochronous OUT endpoint has an incomplete transfer.
2. If this occurs because isochronous OUT data is not completely emptied from the endpoint, the application must ensure that the application empties all isochronous OUT data (data and status) from the receive FIFO before proceeding.
  - When all data are emptied from the receive FIFO, the application can detect the XFRC interrupt (OTG\_DOEPINTx). In this case, the application must re-enable the endpoint to receive isochronous OUT data in the next frame.
3. When it receives an INCOMPISOOUT interrupt (in OTG\_GINTSTS), the application must read the control registers of all isochronous OUT endpoints (OTG\_DOEPCTLx) to determine which endpoints had an incomplete transfer in the current microframe. An endpoint transfer is incomplete if both the following conditions are met:
  - EONUM bit (in OTG\_DOEPCTLx) = FNSOF[0] (in OTG\_DSTS)
  - EPENA = 1 (in OTG\_DOEPCTLx)
4. The previous step must be performed before the SOF interrupt (in OTG\_GINTSTS) is detected, to ensure that the current frame number is not changed.
5. For isochronous OUT endpoints with incomplete transfers, the application must discard the data in the memory and disable the endpoint by setting the EPDIS bit in OTG\_DOEPCTLx.
6. Wait for the EPDISD interrupt (in OTG\_DOEPINTx) and enable the endpoint to receive new data in the next frame.
  - Because the core can take some time to disable the endpoint, the application may not be able to receive the data in the next frame after receiving bad isochronous data.

- **Stalling a non-isochronous OUT endpoint**

This section describes how the application can stall a non-isochronous endpoint.

1. Put the core in the Global OUT NAK mode.
2. Disable the required endpoint
  - When disabling the endpoint, instead of setting the SNAK bit in OTG\_DOEPCTL, set STALL = 1 (in OTG\_DOEPCTL).  
The STALL bit always takes precedence over the NAK bit.
3. When the application is ready to end the STALL handshake for the endpoint, the STALL bit (in OTG\_DOEPCTLx) must be cleared.
4. If the application is setting or clearing a STALL for an endpoint due to a SetFeature.Endpoint Halt or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the status stage transfer on the control endpoint.

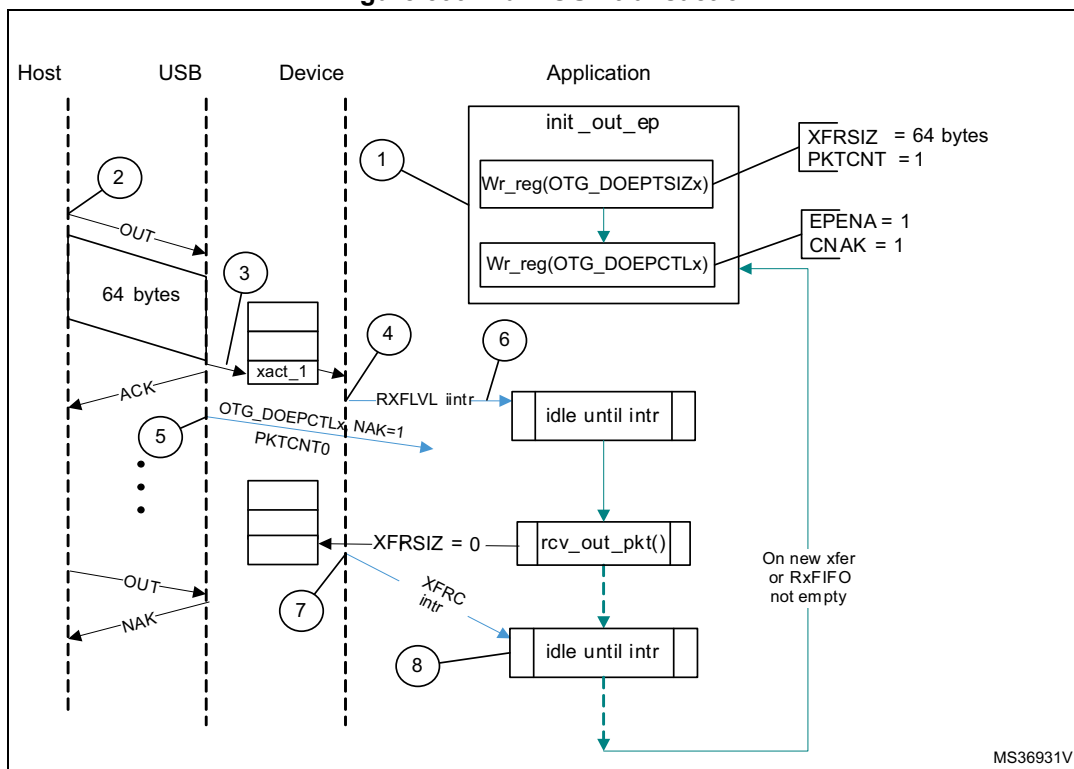
### Examples

This section describes and depicts some fundamental transfer types and scenarios.

- Bulk OUT transaction

*Figure 599* depicts the reception of a single Bulk OUT data packet from the USB to the AHB and describes the events involved in the process.

Figure 599. Bulk OUT transaction



After a SetConfiguration/SetInterface command, the application initializes all OUT endpoints by setting CNAK = 1 and EPENA = 1 (in OTG\_DOEPTLx), and setting a suitable XFRSIZ and PKTCNT in the OTG\_DOEPTSIZx register.

1. host attempts to send data (OUT token) to an endpoint.
2. When the core receives the OUT token on the USB, it stores the packet in the Rx FIFO because space is available there.
3. After writing the complete packet in the Rx FIFO, the core then asserts the RXFLVL interrupt (in OTG\_GINTSTS).
4. On receiving the PKTCNT number of USB packets, the core internally sets the NAK bit for this endpoint to prevent it from receiving any more packets.
5. The application processes the interrupt and reads the data from the Rx FIFO.
6. When the application has read all the data (equivalent to XFRSIZ), the core generates an XFRSIZ interrupt (in OTG\_DOEPINTx).
7. The application processes the interrupt and uses the setting of the XFRSIZ interrupt bit (in OTG\_DOEPINTx) to determine that the intended transfer is complete.

**IN data transfers**

- **Packet write**

This section describes how the application writes data packets to the endpoint FIFO when dedicated transmit FIFOs are enabled.

1. The application can either choose the polling or the interrupt mode.
  - In polling mode, the application monitors the status of the endpoint transmit data FIFO by reading the OTG\_DTXFSTSx register, to determine if there is enough space in the data FIFO.
  - In interrupt mode, the application waits for the TXFE interrupt (in OTG\_DIEPINTx) and then reads the OTG\_DTXFSTSx register, to determine if there is enough space in the data FIFO.
  - To write a single non-zero length data packet, there must be space to write the entire packet in the data FIFO.
  - To write zero length packet, the application must not look at the FIFO space.
2. Using one of the above mentioned methods, when the application determines that there is enough space to write a transmit packet, the application must first write into the endpoint control register, before writing the data into the data FIFO. Typically, the application, must do a read modify write on the OTG\_DIEPCTLx register to avoid modifying the contents of the register, except for setting the endpoint enable bit.

The application can write multiple packets for the same endpoint into the transmit FIFO, if space is available. For periodic IN endpoints, the application must write packets only for one microframe. It can write packets for the next periodic transaction only after getting transfer complete for the previous transaction.

- **Setting IN endpoint NAK**

Internal data flow:

1. When the application sets the IN NAK for a particular endpoint, the core stops transmitting data on the endpoint, irrespective of data availability in the endpoint's transmit FIFO.
2. Non-isochronous IN tokens receive a NAK handshake reply
  - Isochronous IN tokens receive a zero-data-length packet reply
3. The core asserts the INEPNE (IN endpoint NAK effective) interrupt in OTG\_DIEPINTx in response to the SNAK bit in OTG\_DIEPCTLx.
4. Once this interrupt is seen by the application, the application can assume that the endpoint is in IN NAK mode. This interrupt can be cleared by the application by setting the CNAK bit in OTG\_DIEPCTLx.

Application programming sequence:

1. To stop transmitting any data on a particular IN endpoint, the application must set the IN NAK bit. To set this bit, the following field must be programmed.
  - SNAK = 1 in OTG\_DIEPCTLx
2. Wait for assertion of the INEPNE interrupt in OTG\_DIEPINTx. This interrupt indicates that the core has stopped transmitting data on the endpoint.
3. The core can transmit valid IN data on the endpoint after the application has set the NAK bit, but before the assertion of the NAK Effective interrupt.
4. The application can mask this interrupt temporarily by writing to the INEPNEM bit in OTG\_DIEPMSK.
  - INEPNEM = 0 in OTG\_DIEPMSK
5. To exit endpoint NAK mode, the application must clear the NAK status bit (NAKSTS) in OTG\_DIEPCTLx. This also clears the INEPNE interrupt (in OTG\_DIEPINTx).

- CNAK = 1 in OTG\_DIEPCTLx
- 6. If the application masked this interrupt earlier, it must be unmasked as follows:
  - INEPNEM = 1 in OTG\_DIEPMSK

- **IN endpoint disable**

Use the following sequence to disable a specific IN endpoint that has been previously enabled.

Application programming sequence:

1. The application must stop writing data on the AHB for the IN endpoint to be disabled.
2. The application must set the endpoint in NAK mode.
  - SNAK = 1 in OTG\_DIEPCTLx
3. Wait for the INEPNE interrupt in OTG\_DIEPINTx.
4. Set the following bits in the OTG\_DIEPCTLx register for the endpoint that must be disabled.
  - EPDIS = 1 in OTG\_DIEPCTLx
  - SNAK = 1 in OTG\_DIEPCTLx
5. Assertion of the EPDISD interrupt in OTG\_DIEPINTx indicates that the core has completely disabled the specified endpoint. Along with the assertion of the interrupt, the core also clears the following bits:
  - EPENA = 0 in OTG\_DIEPCTLx
  - EPDIS = 0 in OTG\_DIEPCTLx
6. The application must read the OTG\_DIEPTSIZx register for the periodic IN EP, to calculate how much data on the endpoint were transmitted on the USB.
7. The application must flush the data in the endpoint transmit FIFO, by setting the following fields in the OTG\_GRSTCTL register:
  - TXFNUM (in OTG\_GRSTCTL) = Endpoint transmit FIFO number
  - TXFFLSH in (OTG\_GRSTCTL) = 1

The application must poll the OTG\_GRSTCTL register, until the TXFFLSH bit is cleared by the core, which indicates the end of flush operation. To transmit new data on this endpoint, the application can re-enable the endpoint at a later point.

- **Generic non-periodic IN data transfers**

Application requirements:

1. Before setting up an IN transfer, the application must ensure that all data to be transmitted as part of the IN transfer are part of a single buffer.
2. For IN transfers, the transfer size field in the endpoint transfer size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
  - To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:
 
$$\text{Transfer size}[\text{EPNUM}] = x \times \text{MPSIZ}[\text{EPNUM}] + \text{sp}$$
 If (sp > 0), then packet count[EPNUM] = x + 1.  
 Otherwise, packet count[EPNUM] = x
  - To transmit a single zero-length data packet:

Transfer size[EPNUM] = 0

Packet count[EPNUM] = 1

- To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer into two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.
    - First transfer: transfer size[EPNUM] =  $x \times \text{MPSIZ}[\text{epnum}]$ ; packet count =  $n$ ;
    - Second transfer: transfer size[EPNUM] = 0; packet count = 1;
3. Once an endpoint is enabled for data transfers, the core updates the transfer size register. At the end of the IN transfer, the application must read the transfer size register to determine how much data posted in the transmit FIFO have already been sent on the USB.
  4. Data fetched into transmit FIFO = Application-programmed initial transfer size – core-updated final transfer size
    - Data transmitted on USB = (application-programmed initial packet count – core updated final packet count)  $\times$  MPSIZ[EPNUM]
    - Data yet to be transmitted on USB = (Application-programmed initial transfer size – data transmitted on USB)

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the transmit FIFO for the endpoint.
3. Every time a packet is written into the transmit FIFO by the application, the transfer size for that endpoint is decremented by the packet size. The data is fetched from the memory by the application, until the transfer size for the endpoint becomes 0. After writing the data into the FIFO, the “number of packets in FIFO” count is incremented (this is a 3-bit count, internally maintained by the core for each IN endpoint transmit FIFO. The maximum number of packets maintained by the core at any time in an IN endpoint FIFO is eight). For zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.
4. Once the data are written to the transmit FIFO, the core reads them out upon receiving an IN token. For every non-isochronous IN data packet transmitted with an ACK handshake, the packet count for the endpoint is decremented by one, until the packet count is zero. The packet count is not decremented on a timeout.
5. For zero length packets (indicated by an internal zero length flag), the core sends out a zero-length packet for the IN token and decrements the packet count field.
6. If there are no data in the FIFO for a received IN token and the packet count field for that endpoint is zero, the core generates an “IN token received when Tx FIFO is empty” (ITTXFE) interrupt for the endpoint, provided that the endpoint NAK bit is not set. The core responds with a NAK handshake for non-isochronous endpoints on the USB.
7. The core internally rewinds the FIFO pointers and no timeout interrupt is generated.
8. When the transfer size is 0 and the packet count is 0, the transfer complete (XFRC) interrupt for the endpoint is generated and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG\_DIEPTISIZx register with the transfer size and corresponding packet count.
2. Program the OTG\_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA (endpoint enable) bits.
3. When transmitting non-zero length data packet, the application must poll the OTG\_DTXFSTSx register (where x is the FIFO number associated with that endpoint) to determine whether there is enough space in the data FIFO. The application can optionally use TXFE (in OTG\_DIEPINTx) before writing the data.

- **Generic periodic IN data transfers**

This section describes a typical periodic IN data transfer.

Application requirements:

1. Application requirements 1, 2, 3, and 4 of [Generic non-periodic IN data transfers on page 2006](#) also apply to periodic IN data transfers, except for a slight modification of requirement 2.
  - The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To transmit a few maximum-packet-size packets and a short packet at the end of the transfer, the following conditions must be met:
 
$$\text{transfer size[EPNUM]} = x \times \text{MPSIZ[EPNUM]} + \text{sp}$$
 (where x is an integer  $\geq 0$ , and  $0 \leq \text{sp} < \text{MPSIZ[EPNUM]}$ )  
 If ( $\text{sp} > 0$ ),  $\text{packet count[EPNUM]} = x + 1$   
 Otherwise,  $\text{packet count[EPNUM]} = x$ ;  
 $\text{MCNT[EPNUM]} = \text{packet count[EPNUM]}$
  - The application cannot transmit a zero-length data packet at the end of a transfer. It can transmit a single zero-length data packet by itself. To transmit a single zero-length data packet:
    - $\text{transfer size[EPNUM]} = 0$
    - $\text{packet count[EPNUM]} = 1$
    - $\text{MCNT[EPNUM]} = \text{packet count[EPNUM]}$
2. The application can only schedule data transfers one frame at a time.
  - $(\text{MCNT} - 1) \times \text{MPSIZ} \leq \text{XFERSIZ} \leq \text{MCNT} \times \text{MPSIZ}$
  - $\text{PKTCNT} = \text{MCNT}$  (in OTG\_DIEPTISIZx)
  - If  $\text{XFERSIZ} < \text{MCNT} \times \text{MPSIZ}$ , the last data packet of the transfer is a short packet.
  - Note that: MCNT is in OTG\_DIEPTISIZx, MPSIZ is in OTG\_DIEPCTLx, PKTCNT is in OTG\_DIEPTISIZx and XFERSIZ is in OTG\_DIEPTISIZx
3. The complete data to be transmitted in the frame must be written into the transmit FIFO by the application, before the IN token is received. Even when 1 word of the data to be transmitted per frame is missing in the transmit FIFO when the IN token is received, the core behaves as when the FIFO is empty. When the transmit FIFO is empty:
  - A zero data length packet would be transmitted on the USB for isochronous IN endpoints
  - A NAK handshake would be transmitted on the USB for interrupt IN endpoints

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the associated transmit FIFO for the endpoint.
3. Every time the application writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data are fetched from application memory until the transfer size for the endpoint becomes 0.
4. When an IN token is received for a periodic endpoint, the core transmits the data in the FIFO, if available. If the complete data payload (complete packet, in dedicated FIFO mode) for the frame is not present in the FIFO, then the core generates an IN token received when Tx FIFO empty interrupt for the endpoint.
  - A zero-length data packet is transmitted on the USB for isochronous IN endpoints
  - A NAK handshake is transmitted on the USB for interrupt IN endpoints
5. The packet count for the endpoint is decremented by 1 under the following conditions:
  - For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
  - For interrupt endpoints, when an ACK handshake is transmitted
  - When the transfer size and packet count are both 0, the transfer completed interrupt for the endpoint is generated and the endpoint enable is cleared.
6. At the “Periodic frame Interval” (controlled by PFIVL in OTG\_DCFG), when the core finds non-empty any of the isochronous IN endpoint FIFOs scheduled for the current frame non-empty, the core generates an IISOIXFR interrupt in OTG\_GINTSTS.

Application programming sequence:

1. Program the OTG\_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA bits.
2. Write the data to be transmitted in the next frame to the transmit FIFO.
3. Asserting the ITTXFE interrupt (in OTG\_DIEPINTx) indicates that the application has not yet written all data to be transmitted to the transmit FIFO.
4. If the interrupt endpoint is already enabled when this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint so that the data can be transmitted on the next IN token attempt.
5. Asserting the XFRC interrupt (in OTG\_DIEPINTx) with no ITTXFE interrupt in OTG\_DIEPINTx indicates the successful completion of an isochronous IN transfer. A read to the OTG\_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
6. Asserting the XFRC interrupt (in OTG\_DIEPINTx), with or without the ITTXFE interrupt (in OTG\_DIEPINTx), indicates the successful completion of an interrupt IN transfer. A read to the OTG\_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
7. Asserting the incomplete isochronous IN transfer (IISOIXFR) interrupt in OTG\_GINTSTS with none of the aforementioned interrupts indicates the core did not receive at least 1 periodic IN token in the current frame.

- **Incomplete isochronous IN data transfers**

This section describes what the application must do on an incomplete isochronous IN data transfer.



Internal data flow:

1. An isochronous IN transfer is treated as incomplete in one of the following conditions:
  - a) The core receives a corrupted isochronous IN token on at least one isochronous IN endpoint. In this case, the application detects an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG\_GINTSTS).
  - b) The application is slow to write the complete data payload to the transmit FIFO and an IN token is received before the complete data payload is written to the FIFO. In this case, the application detects an IN token received when Tx FIFO empty interrupt in OTG\_DIEPINTx. The application can ignore this interrupt, as it eventually results in an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG\_GINTSTS) at the end of periodic frame.  
  
The core transmits a zero-length data packet on the USB in response to the received IN token.
2. The application must stop writing the data payload to the transmit FIFO as soon as possible.
3. The application must set the NAK bit and the disable bit for the endpoint.
4. The core disables the endpoint, clears the disable bit, and asserts the endpoint disable interrupt for the endpoint.

Application programming sequence:

1. The application can ignore the IN token received when Tx FIFO empty interrupt in OTG\_DIEPINTx on any isochronous IN endpoint, as it eventually results in an incomplete isochronous IN transfer interrupt (in OTG\_GINTSTS).
2. Assertion of the incomplete isochronous IN transfer interrupt (in OTG\_GINTSTS) indicates an incomplete isochronous IN transfer on at least one of the isochronous IN endpoints.
3. The application must read the endpoint control register for all isochronous IN endpoints to detect endpoints with incomplete IN data transfers.
4. The application must stop writing data to the Periodic Transmit FIFOs associated with these endpoints on the AHB.
5. Program the following fields in the OTG\_DIEPCTLx register to disable the endpoint:
  - SNAK = 1 in OTG\_DIEPCTLx
  - EPDIS = 1 in OTG\_DIEPCTLx
6. The assertion of the endpoint disabled interrupt in OTG\_DIEPINTx indicates that the core has disabled the endpoint.
  - At this point, the application must flush the data in the associated transmit FIFO or overwrite the existing data in the FIFO by enabling the endpoint for a new transfer in the next microframe. To flush the data, the application must use the OTG\_GRSTCTL register.

- **Stalling non-isochronous IN endpoints**

This section describes how the application can stall a non-isochronous endpoint.

Application programming sequence:



1. Disable the IN endpoint to be stalled. Set the STALL bit as well.
2. EPDIS = 1 in OTG\_DIEPCTLx, when the endpoint is already enabled
  - STALL = 1 in OTG\_DIEPCTLx
  - The STALL bit always takes precedence over the NAK bit
3. Assertion of the endpoint disabled interrupt (in OTG\_DIEPINTx) indicates to the application that the core has disabled the specified endpoint.
4. The application must flush the non-periodic or periodic transmit FIFO, depending on the endpoint type. In case of a non-periodic endpoint, the application must re-enable the other non-periodic endpoints that do not need to be stalled, to transmit data.
5. Whenever the application is ready to end the STALL handshake for the endpoint, the STALL bit must be cleared in OTG\_DIEPCTLx.
6. If the application sets or clears a STALL bit for an endpoint due to a SetFeature.Endpoint Halt command or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the status stage transfer on the control endpoint.

Special case: stalling the control OUT endpoint

The core must stall IN/OUT tokens if, during the data stage of a control transfer, the host sends more IN/OUT tokens than are specified in the SETUP packet. In this case, the application must enable the ITTXFE interrupt in OTG\_DIEPINTx and the OTEPDIS interrupt in OTG\_DOEPINTx during the data stage of the control transfer, after the core has transferred the amount of data specified in the SETUP packet. Then, when the application receives this interrupt, it must set the STALL bit in the corresponding endpoint control register, and clear this interrupt.

### 50.16.6 Worst case response time

When the OTG\_FS controller acts as a device, there is a worst case response time for any tokens that follow an isochronous OUT. This worst case response time depends on the AHB clock frequency.

The core registers are in the AHB domain, and the core does not accept another token before updating these register values. The worst case is for any token following an isochronous OUT, because for an isochronous transaction, there is no handshake and the next token could come sooner. This worst case value is 7 PHY clocks when the AHB clock is the same as the PHY clock. When the AHB clock is faster, this value is smaller.

If this worst case condition occurs, the core responds to bulk/interrupt tokens with a NAK and drops isochronous and SETUP tokens. The host interprets this as a timeout condition for SETUP and retries the SETUP packet. For isochronous transfers, the Incomplete isochronous IN transfer interrupt (IISOIXFR) and Incomplete isochronous OUT transfer interrupt (IISOOXFR) inform the application that isochronous IN/OUT packets were dropped.

#### Choosing the value of TRDT in OTG\_GUSBCFG

The value in TRDT (OTG\_GUSBCFG) is the time it takes for the MAC, in terms of PHY clocks after it has received an IN token, to get the FIFO status, and thus the first data from the PFC block. This time involves the synchronization delay between the PHY and AHB clocks. The worst case delay for this is when the AHB clock is the same as the PHY clock. In this case, the delay is 5 clocks.

Once the MAC receives an IN token, this information (token received) is synchronized to the AHB clock by the PFC (the PFC runs on the AHB clock). The PFC then reads the data from the SPRAM and writes them into the dual clock source buffer. The MAC then reads the data out of the source buffer (4 deep).

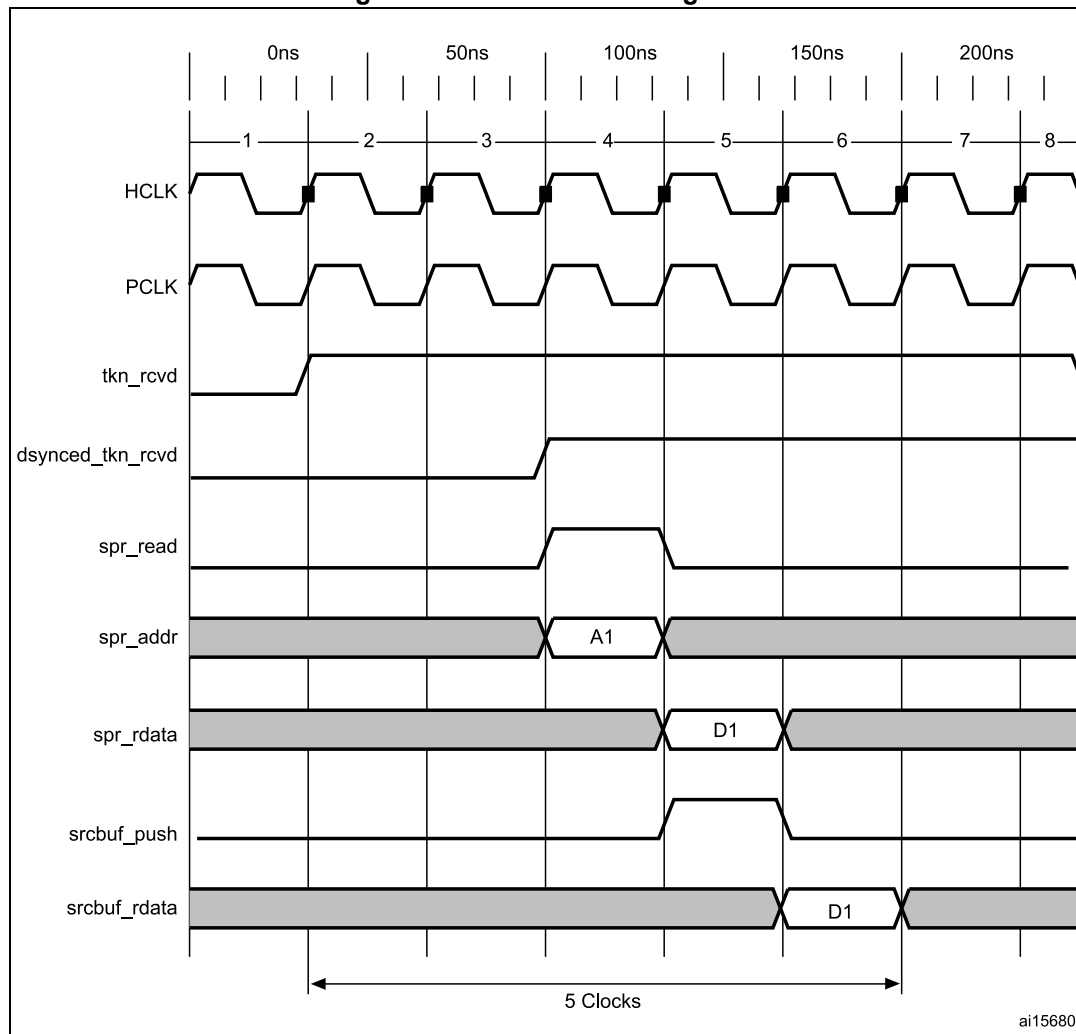
If the AHB is running at a higher frequency than the PHY, the application can use a smaller value for TRDT (in OTG\_GUSBCFG).

Figure 600 has the following signals:

- tkn\_rcvd: Token received information from MAC to PFC
- dynced\_tkn\_rcvd: Doubled sync tkn\_rcvd, from PCLK to HCLK domain
- spr\_read: Read to SPRAM
- spr\_addr: Address to SPRAM
- spr\_rdata: Read data from SPRAM
- srcbuf\_push: Push to the source buffer
- srcbuf\_rdata: Read data from the source buffer. Data seen by MAC

To calculate the value of TRDT, refer to [Table 327: TRDT values \(FS\)](#) .

Figure 600. TRDT max timing case



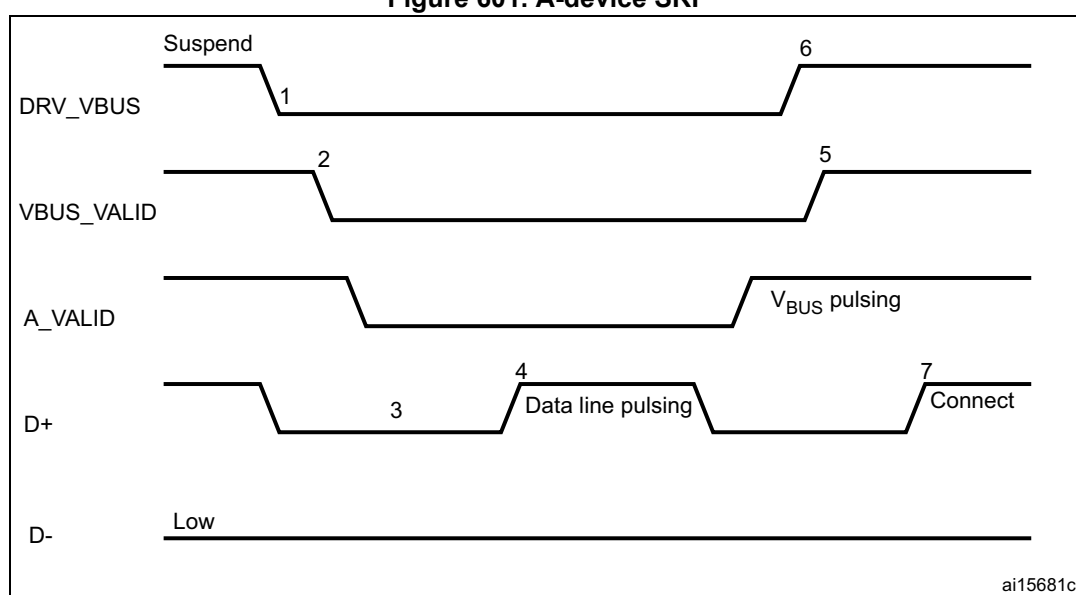
### 50.16.7 OTG programming model

The OTG\_FS controller is an OTG device supporting HNP and SRP. When the core is connected to an “A” plug, it is referred to as an A-device. When the core is connected to a “B” plug it is referred to as a B-device. In host mode, the OTG\_FS controller turns off  $V_{BUS}$  to conserve power. SRP is a method by which the B-device signals the A-device to turn on  $V_{BUS}$  power. A device must perform both data-line pulsing and  $V_{BUS}$  pulsing, but a host can detect either data-line pulsing or  $V_{BUS}$  pulsing for SRP. HNP is a method by which the B-device negotiates and switches to host role. In Negotiated mode after HNP, the B-device suspends the bus and reverts to the device role.

#### A-device session request protocol

The application must set the SRP-capable bit in the core USB configuration register. This enables the OTG\_FS controller to detect SRP as an A-device.

Figure 601. A-device SRP



1. DRV\_VBUS =  $V_{BUS}$  drive signal to the PHY  
 VBUS\_VALID =  $V_{BUS}$  valid signal from PHY  
 A\_VALID = A-peripheral  $V_{BUS}$  level signal to PHY  
 D+ = Data plus line  
 D- = Data minus line

The following points refer and describe the signal numeration shown in the [Figure 601](#):

1. To save power, the application suspends and turns off port power when the bus is idle by writing the port suspend and port power bits in the host port control and status register.
2. PHY indicates port power off by deasserting the VBUS\_VALID signal.
3. The device must detect SE0 for at least 2 ms to start SRP when  $V_{BUS}$  power is off.
4. To initiate SRP, the device turns on its data line pull-up resistor for 5 to 10 ms. The OTG\_FS controller detects data-line pulsing.
5. The device drives  $V_{BUS}$  above the A-device session valid (2.0 V minimum) for  $V_{BUS}$  pulsing.  
 The OTG\_FS controller interrupts the application on detecting SRP. The session

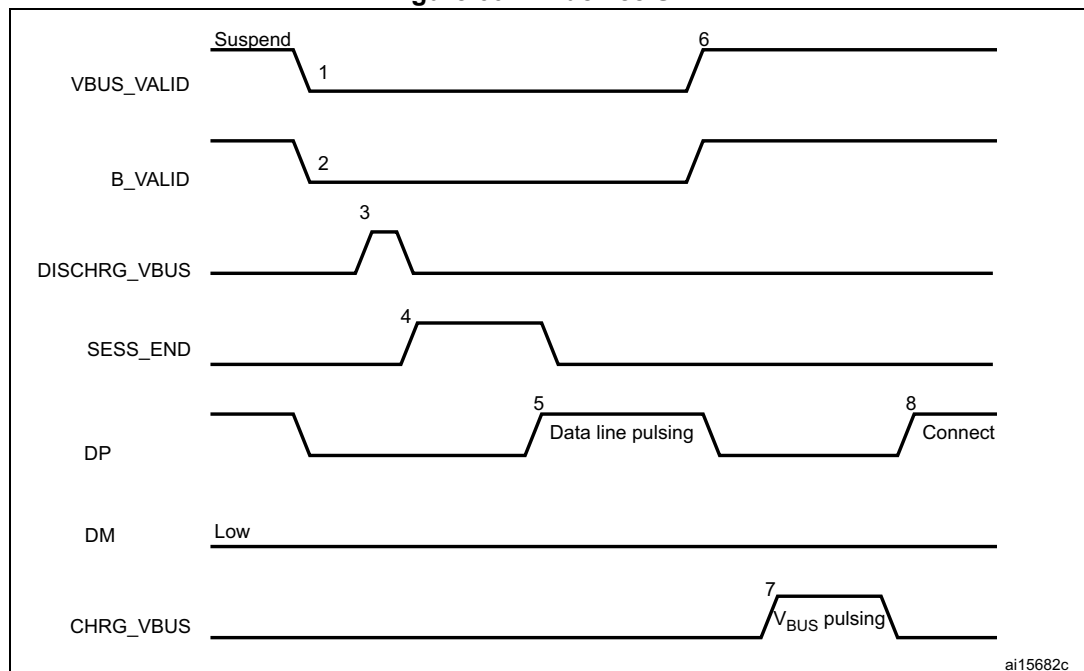
request detected bit is set in Global interrupt status register (SRQINT set in OTG\_GINTSTS).

6. The application must service the session request detected interrupt and turn on the port power bit by writing the port power bit in the host port control and status register. The PHY indicates port power-on by asserting the VBUS\_VALID signal.
7. When the USB is powered, the device connects, completing the SRP process.

**B-device session request protocol**

The application must set the SRP-capable bit in the core USB configuration register. This enables the OTG\_FS controller to initiate SRP as a B-device. SRP is a means by which the OTG\_FS controller can request a new session from the host.

**Figure 602. B-device SRP**



1. VBUS\_VALID = V<sub>BUS</sub> valid signal from PHY  
 B\_VALID = B-peripheral valid session to PHY  
 DISCHRG\_VBUS = discharge signal to PHY  
 SESS\_END = session end signal to PHY  
 CHRГ\_VBUS = charge V<sub>BUS</sub> signal to PHY  
 DP = Data plus line  
 DM = Data minus line

The following points refer and describe the signal numeration shown in the [Figure 602](#):

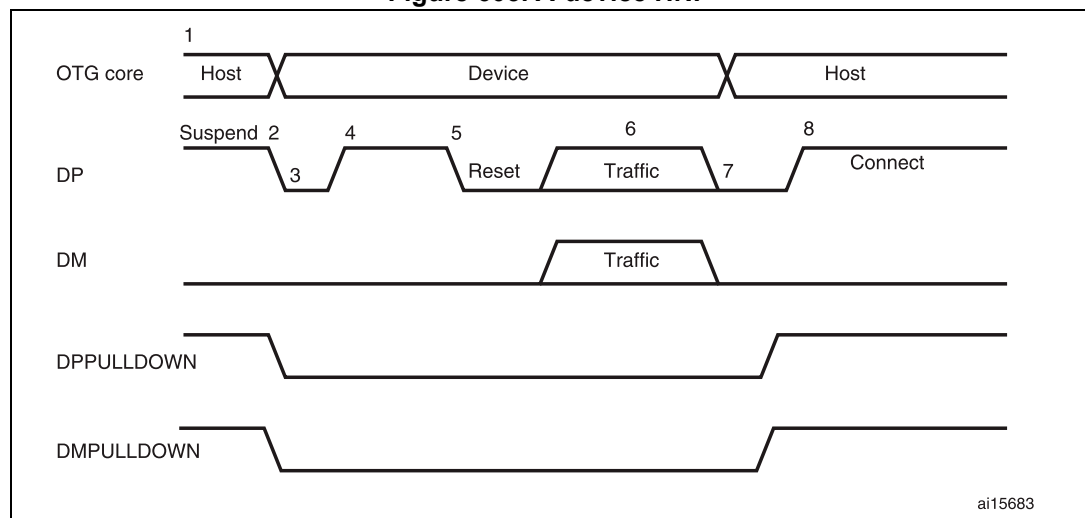
1. To save power, the host suspends and turns off port power when the bus is idle. The OTG\_FS controller sets the early suspend bit in the core interrupt register after 3 ms of bus idleness. Following this, the OTG\_FS controller sets the USB suspend bit in the core interrupt register. The OTG\_FS controller informs the PHY to discharge V<sub>BUS</sub>.
2. The PHY indicates the session's end to the device. This is the initial condition for SRP. The OTG\_FS controller requires 2 ms of SE0 before initiating SRP. For a USB 1.1 full-speed serial transceiver, the application must wait until V<sub>BUS</sub> discharges to 0.2 V after BSVLD (in OTG\_GOTGCTL) is deasserted. This discharge

- time can be obtained from the transceiver vendor and varies from one transceiver to another.
3. The OTG\_FS core informs the PHY to speed up  $V_{BUS}$  discharge.
  4. The application initiates SRP by writing the session request bit in the OTG control and status register. The OTG\_FS controller perform data-line pulsing followed by  $V_{BUS}$  pulsing.
  5. The host detects SRP from either the data-line or  $V_{BUS}$  pulsing, and turns on  $V_{BUS}$ . The PHY indicates  $V_{BUS}$  power-on to the device.
  6. The OTG\_FS controller performs  $V_{BUS}$  pulsing. The host starts a new session by turning on  $V_{BUS}$ , indicating SRP success. The OTG\_FS controller interrupts the application by setting the session request success status change bit in the OTG interrupt status register. The application reads the session request success bit in the OTG control and status register.
  7. When the USB is powered, the OTG\_FS controller connects, completing the SRP process.

**A-device host negotiation protocol**

HNP switches the USB host role from the A-device to the B-device. The application must set the HNP-capable bit in the core USB configuration register to enable the OTG\_FS controller to perform HNP as an A-device.

**Figure 603. A-device HNP**



1. DPPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DP line inside the PHY.  
DMPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DM line inside the PHY.

The following points refer and describe the signal numeration shown in the [Figure 603](#):

1. The OTG\_FS controller sends the B-device a SetFeature b\_hnp\_enable descriptor to enable HNP support. The B-device's ACK response indicates that the B-device supports HNP. The application must set host Set HNP enable bit in the OTG control

and status register to indicate to the OTG\_FS controller that the B-device supports HNP.

2. When it has finished using the bus, the application suspends by writing the port suspend bit in the host port control and status register.
3. When the B-device observes a USB suspend, it disconnects, indicating the initial condition for HNP. The B-device initiates HNP only when it must switch to the host role; otherwise, the bus continues to be suspended.

The OTG\_FS controller sets the host negotiation detected interrupt in the OTG interrupt status register, indicating the start of HNP.

The OTG\_FS controller deasserts the DM pull down and DM pull down in the PHY to indicate a device role. The PHY enables the OTG\_DP pull-up resistor to indicate a connect for B-device.

The application must read the current mode bit in the OTG control and status register to determine device mode operation.

4. The B-device detects the connection, issues a USB reset, and enumerates the OTG\_FS controller for data traffic.
5. The B-device continues the host role, initiating traffic, and suspends the bus when done.

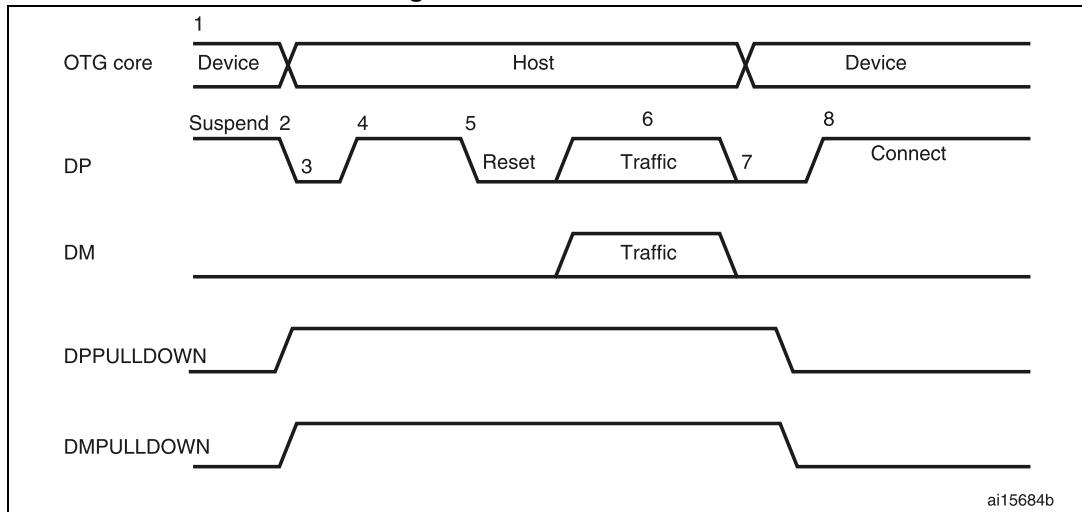
The OTG\_FS controller sets the early suspend bit in the core interrupt register after 3 ms of bus idleness. Following this, the OTG\_FS controller sets the USB suspend bit in the core interrupt register.

6. In Negotiated mode, the OTG\_FS controller detects the suspend, disconnects, and switches back to the host role. The OTG\_FS controller asserts the DM pull down and DM pull down in the PHY to indicate its assumption of the host role.
7. The OTG\_FS controller sets the connector ID status change interrupt in the OTG interrupt status register. The application must read the connector ID status in the OTG control and status register to determine the OTG\_FS controller operation as an A-device. This indicates the completion of HNP to the application. The application must read the Current mode bit in the OTG control and status register to determine host mode operation.
8. The B-device connects, completing the HNP process.

### **B-device host negotiation protocol**

HNP switches the USB host role from B-device to A-device. The application must set the HNP-capable bit in the core USB configuration register to enable the OTG\_FS controller to perform HNP as a B-device.

Figure 604. B-device HNP



1. DPPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DP line inside the PHY.  
DMPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DM line inside the PHY.

The following points refer and describe the signal numeration shown in the [Figure 604](#):

1. The A-device sends the SetFeature b\_hnp\_enable descriptor to enable HNP support. The OTG\_FS controller's ACK response indicates that it supports HNP. The application must set the device HNP enable bit in the OTG control and status register to indicate HNP support.

The application sets the HNP request bit in the OTG control and status register to indicate to the OTG\_FS controller to initiate HNP.

2. When it has finished using the bus, the A-device suspends by writing the port suspend bit in the host port control and status register.

The OTG\_FS controller sets the Early suspend bit in the core interrupt register after 3 ms of bus idleness. Following this, the OTG\_FS controller sets the USB suspend bit in the core interrupt register.

The OTG\_FS controller disconnects and the A-device detects SE0 on the bus, indicating HNP. The OTG\_FS controller asserts the DP pull down and DM pull down in the PHY to indicate its assumption of the host role.

The A-device responds by activating its OTG\_DP pull-up resistor within 3 ms of detecting SE0. The OTG\_FS controller detects this as a connect.

The OTG\_FS controller sets the host negotiation success status change interrupt in the OTG interrupt status register, indicating the HNP status. The application must read the host negotiation success bit in the OTG control and status register to determine host

negotiation success. The application must read the current Mode bit in the core interrupt register (OTG\_GINTSTS) to determine host mode operation.

3. The application sets the reset bit (PRST in OTG\_HPRT) and the OTG\_FS controller issues a USB reset and enumerates the A-device for data traffic.
4. The OTG\_FS controller continues the host role of initiating traffic, and when done, suspends the bus by writing the port suspend bit in the host port control and status register.
5. In Negotiated mode, when the A-device detects a suspend, it disconnects and switches back to the host role. The OTG\_FS controller deasserts the DP pull down and DM pull down in the PHY to indicate the assumption of the device role.
6. The application must read the current mode bit in the core interrupt (OTG\_GINTSTS) register to determine the host mode operation.
7. The OTG\_FS controller connects, completing the HNP process.



# 51 Debug support (DBG)

## 51.1 Overview

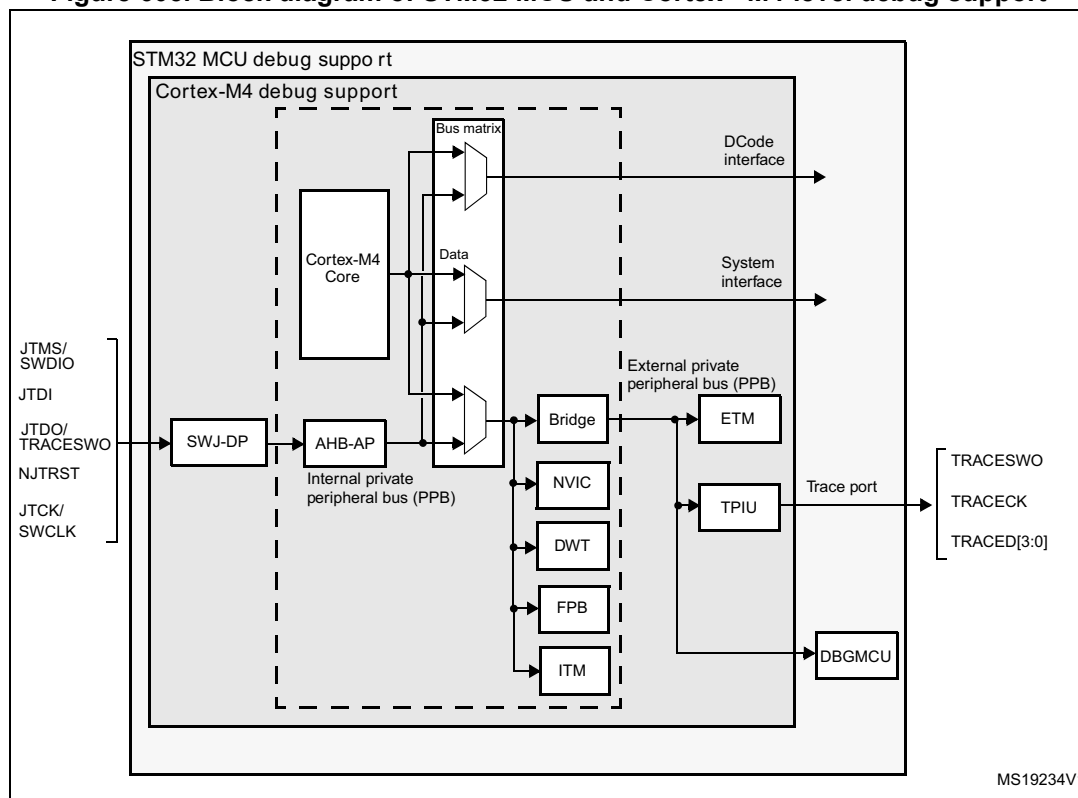
The STM32L4Rxxx and STM32L4Sxxx devices are built around a Cortex<sup>®</sup>-M4 core which contains hardware extensions for advanced debugging features. The debug extensions allow the core to be stopped either on a given instruction fetch (breakpoint) or data access (watchpoint). When stopped, the core's internal state and the system's external state may be examined. Once examination is complete, the core and the system may be restored and program execution resumed.

The debug features are used by the debugger host when connecting to and debugging the STM32L4Rxxx and STM32L4Sxxx MCUs.

Two interfaces for debug are available:

- Serial wire
- JTAG debug port

**Figure 605. Block diagram of STM32 MCU and Cortex<sup>®</sup>-M4-level debug support**



*Note:* The debug features embedded in the Cortex<sup>®</sup>-M4 core are a subset of the Arm<sup>®</sup> CoreSight Design Kit.

The Arm® Cortex®-M4 core provides integrated on-chip debug support. It is comprised of:

- SWJ-DP: Serial wire / JTAG debug port
- AHP-AP: AHB access port
- ITM: Instrumentation trace macrocell
- FPB: Flash patch breakpoint
- DWT: Data watchpoint trigger
- TPUI: Trace port unit interface (available on larger packages, where the corresponding pins are mapped)
- ETM: Embedded Trace Macrocell (available only on STM32L4Rxxx and STM32L4Sxxx devices larger packages, where the corresponding pins are mapped)

It also includes debug features dedicated to the STM32L4Rxxx and STM32L4Sxxx:

- Flexible debug pinout assignment
- MCU debug box (support for low-power modes, control over peripheral clocks, etc.)

*Note:* For further information on debug functionality supported by the Arm® Cortex®-M4 core, refer to the Cortex®-M4-r0p1 Technical Reference Manual and to the CoreSight Design Kit-r0p1 TRM (see [Section 51.2: Reference Arm® documentation](#)).

## 51.2 Reference Arm® documentation

- Cortex®-M4 r0p1 Technical Reference Manual (TRM), search for “Cortex®-M4 Technical Reference Manual” at <http://infocenter.arm.com>
- Arm® Debug Interface V5
- Arm® CoreSight Design Kit revision r0p1 Technical Reference Manual

## 51.3 SWJ debug port (serial wire and JTAG)

The STM32L4Rxxx and STM32L4Sxxx core integrates the Serial Wire / JTAG Debug Port (SWJ-DP). It is an Arm® standard CoreSight debug port that combines a JTAG-DP (5-pin) interface and a SW-DP (2-pin) interface.

- The JTAG Debug Port (JTAG-DP) provides a 5-pin standard JTAG interface to the AHP-AP port.
- The Serial Wire Debug Port (SW-DP) provides a 2-pin (clock + data) interface to the AHP-AP port.

In the SWJ-DP, the two JTAG pins of the SW-DP are multiplexed with some of the five JTAG pins of the JTAG-DP.

Figure 606. SWJ debug port

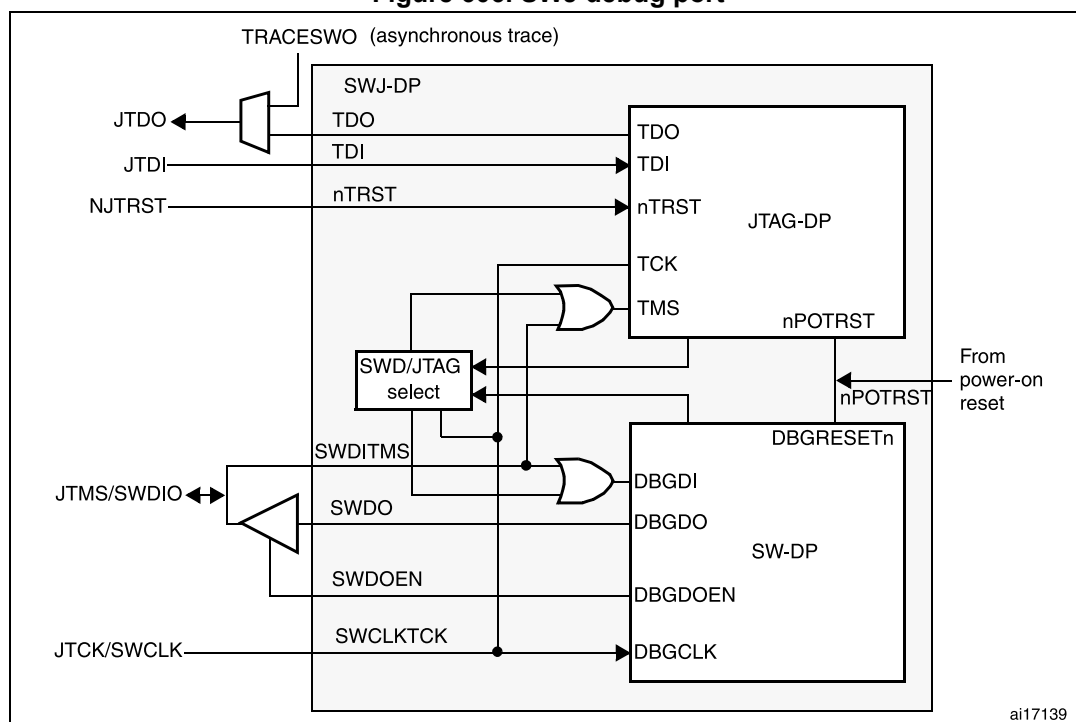


Figure 606 shows that the asynchronous TRACE output (TRACESWO) is multiplexed with TDO. This means that the asynchronous trace can only be used with SW-DP, not JTAG-DP.

### 51.3.1 Mechanism to select the JTAG-DP or the SW-DP

By default, the JTAG-Debug Port is active.

If the debugger host wants to switch to the SW-DP, it must provide a dedicated JTAG sequence on TMS/TCK (respectively mapped to SWDIO and SWCLK) which disables the JTAG-DP and enables the SW-DP. This way it is possible to activate the SWDP using only the SWCLK and SWDIO pins.

This sequence is:

1. Send more than 50 TCK cycles with TMS (SWDIO) = 1
2. Send the 16-bit sequence on TMS (SWDIO) = 0111100111100111 (MSB transmitted first)
3. Send more than 50 TCK cycles with TMS (SWDIO) = 1

## 51.4 Pinout and debug port pins

The STM32L4Rxxx and STM32L4Sxxx MCUs are available in various packages with different numbers of available pins. As a result, some functionalities (ETM) related to pin availability may differ between packages.

### 51.4.1 SWJ debug port pins

Five pins are used as outputs from the STM32L4Rxxx and STM32L4Sxxx for the SWJ-DP as *alternate functions* of general-purpose I/Os. These pins are available on all packages.

**Table 330. SWJ debug port pins**

SWJ-DP pin name	JTAG debug port		SW debug port		Pin assignment
	Type	Description	Type	Debug assignment	
JTMS/SWDIO	I	JTAG Test Mode Selection	IO	Serial Wire Data Input/Output	PA13
JTCK/SWCLK	I	JTAG Test Clock	I	Serial Wire Clock	PA14
JTDI	I	JTAG Test Data Input	-	-	PA15
JTDO/TRACESWO	O	JTAG Test Data Output	-	TRACESWO if asynchronous trace is enabled	PB3
NJTRST	I	JTAG Test nReset	-	-	PB4

### 51.4.2 Flexible SWJ-DP pin assignment

After RESET (SYSRESETn or PORESETn), all five pins used for the SWJ-DP are assigned as dedicated pins immediately usable by the debugger host (note that the trace outputs are not assigned except if explicitly programmed by the debugger host).

However, the STM32L4Rxxx and STM32L4Sxxx MCUs offer the possibility of disabling some or all of the SWJ-DP ports, and therefore the possibility of releasing the associated pins for general-purpose I/O (GPIO) usage, except for NJTRST that can be left disconnected but cannot be used as general purpose GPIO without losing debugger connection. For more details on how to disable SWJ-DP port pins, please refer to [Section 8.3.2: I/O pin alternate function multiplexer and mapping](#).

**Table 331. Flexible SWJ-DP pin assignment**

Available debug ports	SWJ IO pin assigned				
	PA13 / JTMS / SWDIO	PA14 / JTCK / SWCLK	PA15 / JTDI	PB3 / JTDO	PB4 / NJTRST
Full SWJ (JTAG-DP + SW-DP) - Reset State	X	X	X	X	X
Full SWJ (JTAG-DP + SW-DP) but without NJTRST	X	X	X	X	
JTAG-DP disabled and SW-DP enabled	X	X			
JTAG-DP disabled and SW-DP disabled	Released				

*Note:* When the APB bridge write buffer is full, it takes one extra APB cycle when writing the AFIO\_MAPR register. This is because the deactivation of the JTAGSW pins is done in two cycles to guarantee a clean level on the nTRST and TCK input signals of the core.

- Cycle 1: the JTAGSW input signals to the core are tied to 1 or 0 (to 1 for nTRST, TDI and TMS, to 0 for TCK)
- Cycle 2: the GPIO controller takes the control signals of the SWJTAG IO pins (like controls of direction, pull-up/down, Schmitt trigger activation, etc.).

### 51.4.3 Internal pull-up and pull-down on JTAG pins

It is necessary to ensure that the JTAG input pins are not floating since they are directly connected to flip-flops to control the debug mode features. Special care must be taken with the SWCLK/TCK pin which is directly connected to the clock of some of these flip-flops.

To avoid any uncontrolled IO levels, the device embeds internal pull-ups and pull-downs on the JTAG input pins:

- NJTRST: internal pull-up
- JTDI: internal pull-up
- JTMS/SWDIO: internal pull-up
- TCK/SWCLK: internal pull-down

Once a JTAG IO is released by the user software, the GPIO controller takes control again. The reset states of the GPIO control registers put the I/Os in the equivalent state:

- NJTRST: input pull-up
- JTDI: input pull-up
- JTMS/SWDIO: input pull-up
- JTCK/SWCLK: input pull-down
- JTDO: input floating

The software can then use these I/Os as standard GPIOs.

*Note:* The JTAG IEEE standard recommends to add pull-ups on TDI, TMS and nTRST but there is no special recommendation for TCK. However, for JTCK, the device needs an integrated pull-down.

*Having embedded pull-ups and pull-downs removes the need to add external resistors.*

### 51.4.4 Using serial wire and releasing the unused debug pins as GPIOs

To use the serial wire DP to release some GPIOs, the user software must change the GPIO (PA15, PB3 and PB4) configuration mode in the GPIO\_MODER register. This releases PA15, PB3 and PB4 which now become available as GPIOs.

When debugging, the host performs the following actions:

- Under system reset, all SWJ pins are assigned (JTAG-DP + SW-DP).
- Under system reset, the debugger host sends the JTAG sequence to switch from the JTAG-DP to the SW-DP.
- Still under system reset, the debugger sets a breakpoint on vector reset.
- The system reset is released and the Core halts.
- All the debug communications from this point are done using the SW-DP. The other JTAG pins can then be reassigned as GPIOs by the user software.

*Note:* For user software designs, note that:

*To release the debug pins, remember that they will be first configured either in input-pull-up (nTRST, TMS, TDI) or pull-down (TCK) or output tristate (TDO) for a certain duration after reset until the instant when the user software releases the pins.*

*When debug pins (JTAG or SW or TRACE) are mapped, changing the corresponding IO pin configuration in the IOPORT controller has no effect.*

## 51.5 STM32L4Rxxx and STM32L4Sxxx JTAG TAP connection

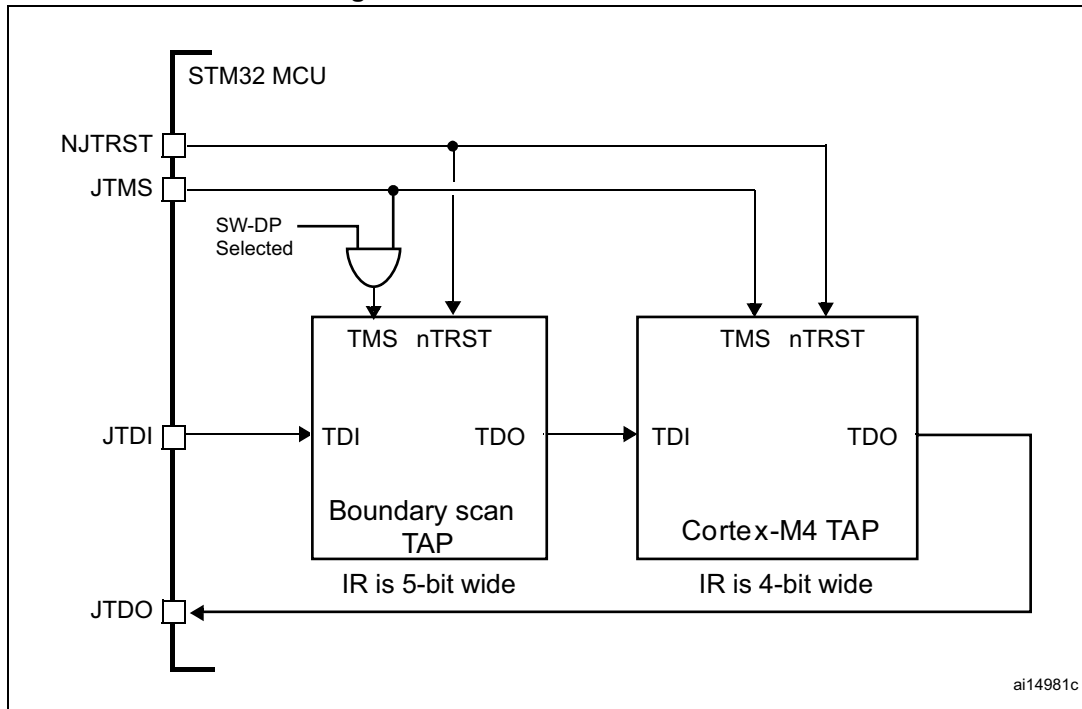
The STM32L4Rxxx and STM32L4Sxxx MCUs integrate two serially connected JTAG TAPs, the boundary scan TAP (IR is 5-bit wide) and the Cortex<sup>®</sup>-M4 TAP (IR is 4-bit wide).

To access the TAP of the Cortex<sup>®</sup>-M4 for debug purposes:

1. First, it is necessary to shift the BYPASS instruction of the boundary scan TAP.
2. Then, for each IR shift, the scan chain contains 9 bits (=5+4) and the unused TAP instruction must be shifted by using the BYPASS instruction.
3. For each data shift, the unused TAP, which is in BYPASS mode, adds 1 extra data bit in the data scan chain.

*Note:* **Important:** Once Serial-Wire is selected using the dedicated Arm<sup>®</sup> JTAG sequence, the boundary scan TAP is automatically disabled (JTMS forced high).

Figure 607. JTAG TAP connections



### 51.6 ID codes and locking mechanism

There are several ID codes inside the STM32L4Rxxx and STM32L4Sxxx MCUs. ST strongly recommends tools designers to lock their debuggers using the MCU DEVICE ID code located in the external PPB memory map at address 0xE0042000.

### 51.6.1 MCU device ID code

The STM32L4Rxxx and STM32L4Sxxx MCUs integrate an MCU ID code. This ID identifies the ST MCU part-number and the die revision. It is part of the DBG\_MCU component and is mapped on the external PPB bus (see [Section 51.16 on page 2038](#)). This code is accessible using the JTAG debug port (4 to 5 pins) or the SW debug port (two pins) or by the user software. It is even accessible while the MCU is under system reset.

Only the DEV\_ID(11:0) should be used for identification by the debugger/programmer tools.

#### DBGMCU\_IDCODE

Address: 0xE004 2000

Only 32-bits access supported. Read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DEV_ID											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV\_ID[15:0]** Revision identifier  
 This field indicates the revision of the device.  
 0x1000: Revision A  
 0x1001: Revision Z  
 0x1003: Revision Y

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV\_ID[11:0]**: Device identifier  
 The device ID is: 0x470

### 51.6.2 Boundary scan TAP

#### JTAG ID code

The TAP of the STM32L4Rxxx and STM32L4Sxxx BSC (boundary scan) integrates a JTAG ID code equal to 0x06470041.

### 51.6.3 Cortex<sup>®</sup>-M4 TAP

The TAP of the Arm<sup>®</sup> Cortex<sup>®</sup>-M4 integrates a JTAG ID code. This ID code is the Arm<sup>®</sup> default one and has not been modified. This code is only accessible by the JTAG Debug Port.

This code is **0x4BA00477** (corresponds to Cortex<sup>®</sup>-M4 r0p1, see [Section 51.2: Reference Arm<sup>®</sup> documentation](#)).

### 51.6.4 Cortex<sup>®</sup>-M4 JEDEC-106 ID code

The Arm<sup>®</sup> Cortex<sup>®</sup>-M4 integrates a JEDEC-106 ID code. It is located in the 4KB ROM table mapped on the internal PPB bus at address 0xE00FF000\_0xE00FFFFF.



This code is accessible by the JTAG Debug Port (4 to 5 pins) or by the SW Debug Port (two pins) or by the user software.

## 51.7 JTAG debug port

A standard JTAG state machine is implemented with a 4-bit instruction register (IR) and five data registers (for full details, refer to the Cortex<sup>®</sup>-M4 with FPU r0p1 *Technical Reference Manual (TRM)*, for references, please see [Section 51.2: Reference Arm<sup>®</sup> documentation](#)).

**Table 332. JTAG debug port data registers**

IR(3:0)	Data register	Details
1111	BYPASS [1 bit]	-
1110	IDCODE [32 bits]	ID CODE 0x3BA00477 (Arm <sup>®</sup> Cortex <sup>®</sup> -M4 r0p1-01rel0 ID Code)
1010	DPACC [35 bits]	<p>Debug port access register This initiates a debug port and allows 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 which is read following a read request</li> <li>Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010 = OK/FAULT 001 = WAIT OTHER = reserved</li> </ul> </li> </ul> <p>Refer to <a href="#">Table 333</a> for a description of the A(3:2) bits</p>
1011	APACC [35 bits]	<p>Access port access register Initiates an access port and allows 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 address (sub-address AP registers).</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 which is read following a read request</li> <li>Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010 = OK/FAULT 001 = WAIT OTHER = reserved</li> </ul> </li> </ul> <p>There are many AP Registers (see AHB-AP) addressed as the combination of:</p> <ul style="list-style-type: none"> <li>– The shifted value A[3:2]</li> <li>– The current value of the DP SELECT register</li> </ul>
1000	ABORT [35 bits]	<p>Abort register</p> <ul style="list-style-type: none"> <li>– Bits 31:1 = Reserved</li> <li>– Bit 0 = DAPABORT: write 1 to generate a DAP abort.</li> </ul>

**Table 333. 32-bit debug port registers addressed through the shifted value A[3:2]**

Address	A(3:2) value	Description
0x0	00	Reserved, must be kept at reset value.
0x4	01	DP CTRL/STAT register. Used to: <ul style="list-style-type: none"> <li>– Request a system or debug power-up</li> <li>– Configure the transfer operation for AP accesses</li> <li>– Control the pushed compare and pushed verify operations</li> <li>– Read some status flags (overrun, power-up acknowledges)</li> </ul>
0x8	10	DP SELECT register. Used to select the current access port and the active 4-words register window. <ul style="list-style-type: none"> <li>– Bits 31:24: APSEL: select the current AP</li> <li>– Bits 23:8: reserved</li> <li>– Bits 7:4: APBANKSEL: select the active 4-words register window on the current AP</li> <li>– Bits 3:0: reserved</li> </ul>
0xC	11	DP RDBUFF register: Used to allow the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation)

## 51.8 SW debug port

### 51.8.1 SW protocol introduction

This synchronous serial protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bidirectional

The protocol allows two banks of registers (DPACC registers and APACC registers) to be read and written to.

Bits are transferred LSB-first on the wire.

For SWDIO bidirectional management, the line must be pulled-up on the board (100 kΩ recommended by Arm®).

Each time the direction of SWDIO changes in the protocol, a turnaround time is inserted where the line is not driven by the host nor the target. By default, this turnaround time is one bit time, however this can be adjusted by configuring the SWCLK frequency.

### 51.8.2 SW protocol sequence

Each sequence consist of three phases:

1. Packet request (8 bits) transmitted by the host
2. Acknowledge response (3 bits) transmitted by the target
3. Data transfer phase (33 bits) transmitted by the host or the target

**Table 334. Packet request (8-bits)**

Bit	Name	Description
0	Start	Must be "1"
1	APnDP	0: DP Access 1: AP Access
2	RnW	0: Write Request 1: Read Request
4:3	A(3:2)	Address field of the DP or AP registers (refer to <a href="#">Table 333</a> )
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by the host. Must be read as "1" by the target because of the pull-up

Refer to the Cortex<sup>®</sup>-M4 r0p1 *TRM* for a detailed description of DPACC and APACC registers.

The packet request is always followed by the turnaround time (default 1 bit) where neither the host nor target drive the line.

**Table 335. ACK response (3 bits)**

Bit	Name	Description
0..2	ACK	001: FAULT 010: WAIT 100: OK

The ACK Response must be followed by a turnaround time only if it is a READ transaction or if a WAIT or FAULT acknowledge has been received.

**Table 336. DATA transfer (33 bits)**

Bit	Name	Description
0..31	WDATA or RDATA	Write or Read data
32	Parity	Single parity of the 32 data bits

The DATA transfer must be followed by a turnaround time only if it is a READ transaction.

### 51.8.3 SW-DP state machine (reset, idle states, ID code)

The State Machine of the SW-DP has an internal ID code which identifies the SW-DP. It follows the JEP-106 standard. This ID code is the default Arm<sup>®</sup> one and is set to **0x1BA01477** (corresponding to Cortex<sup>®</sup>-M4 r0p1).

*Note:* Note that the SW-DP state machine is inactive until the target reads this ID code.

- The SW-DP state machine is in RESET STATE either after power-on reset, or after the DP has switched from JTAG to SWD or after the line is high for more than 50 cycles
- The SW-DP state machine is in IDLE STATE if the line is low for at least two cycles after RESET state.
- After RESET state, it is **mandatory** to first enter into an IDLE state AND to perform a READ access of the DP-SW ID CODE register. Otherwise, the target will issue a FAULT acknowledge response on another transactions.

Further details of the SW-DP state machine can be found in the *Cortex<sup>®</sup>-M4 r0p1 TRM* and the *CoreSight Design Kit r0p1 TRM*.

### 51.8.4 DP and AP read/write accesses

- Read accesses to the DP are not posted: the target response can be immediate (if ACK=OK) or can be delayed (if ACK=WAIT).
- Read accesses to the AP are posted. This means that the result of the access is returned on the next transfer. If the next access to be done is NOT an AP access, then the DP-RDBUFF register must be read to obtain the result. The READOK flag of the DP-CTRL/STAT register is updated on every AP read access or RDBUFF read request to know if the AP read access was successful.
- The SW-DP implements a write buffer (for both DP or AP writes), that enables it to accept a write operation even when other transactions are still outstanding. If the write buffer is full, the target acknowledge response is "WAIT". With the exception of IDCODE read or CTRL/STAT read or ABORT write which are accepted even if the write buffer is full.
- Because of the asynchronous clock domains SWCLK and HCLK, two extra SWCLK cycles are needed after a write transaction (after the parity bit) to make the write effective internally. These cycles should be applied while driving the line low (IDLE state)  
This is particularly important when writing the CTRL/STAT for a power-up request. If the next transaction (requiring a power-up) occurs immediately, it will fail.

### 51.8.5 SW-DP registers

Access to these registers are initiated when APnDP=0

**Table 337. SW-DP registers**

A(3:2)	R/W	CTRLSEL bit of SELECT register	Register	Notes
00	Read	-	IDCODE	The manufacturer code is not set to ST code <b>0x2BA01477</b> (identifies the SW-DP)
00	Write	-	ABORT	-

Table 337. SW-DP registers (continued)

A(3:2)	R/W	CTRLSEL bit of SELECT register	Register	Notes
01	Read/Write	0	DP-CTRL/STAT	<p>Purpose is to:</p> <ul style="list-style-type: none"> <li>– request a system or debug power-up</li> <li>– configure the transfer operation for AP accesses</li> <li>– control the pushed compare and pushed verify operations.</li> <li>– read some status flags (overrun, power-up acknowledges)</li> </ul>
01	Read/Write	1	WIRE CONTROL	Purpose is to configure the physical serial port protocol (like the duration of the turnaround time)
10	Read	-	READ RESEND	Enables recovery of the read data from a corrupted debugger transfer, without repeating the original AP transfer.
10	Write	-	SELECT	The purpose is to select the current access port and the active 4-words register window
11	Read/Write	-	READ BUFFER	<p>This read buffer is useful because AP accesses are posted (the result of a read AP request is available on the next AP transaction).</p> <p>This read buffer captures data from the AP, presented as the result of a previous read, without initiating a new transaction</p>

### 51.8.6 SW-AP registers

Access to these registers are initiated when APnDP=1

There are many AP Registers (see AHB-AP) addressed as the combination of:

- The shifted value A[3:2]
- The current value of the DP SELECT register

## 51.9 AHB-AP (AHB access port) - valid for both JTAG-DP and SW-DP

### Features:

- System access is independent of the processor status.
- Either SW-DP or JTAG-DP accesses AHB-AP.
- The AHB-AP is an AHB master into the Bus Matrix. Consequently, it can access all the data buses (Dcode Bus, System Bus, internal and external PPB bus) but the ICode bus.
- Bitband transactions are supported.
- AHB-AP transactions bypass the FPB.

The address of the 32-bits AHP-AP registers are 6-bits wide (up to 64 words or 256 bytes) and consists of:

- c) Bits [7:4] = the bits [7:4] APBANKSEL of the DP SELECT register
- d) Bits [3:2] = the 2 address bits of A(3:2) of the 35-bit packet request for SW-DP.

The AHB-AP of the Cortex<sup>®</sup>-M4 includes 9 x 32-bits registers:

**Table 338. Cortex<sup>®</sup>-M4 AHB-AP registers**

Address offset	Register name	Notes
0x00	AHB-AP Control and Status Word	Configures and controls transfers through the AHB interface (size, hprot, status on current transfer, address increment type)
0x04	AHB-AP Transfer Address	-
0x0C	AHB-AP Data Read/Write	-
0x10	AHB-AP Banked Data 0	Directly maps the 4 aligned data words without rewriting the Transfer Address Register.
0x14	AHB-AP Banked Data 1	
0x18	AHB-AP Banked Data 2	
0x1C	AHB-AP Banked Data 3	
0xF8	AHB-AP Debug ROM Address	Base Address of the debug interface
0xFC	AHB-AP ID Register	-

Refer to the Cortex<sup>®</sup>-M4 *r0p1 TRM* for further details.

## 51.10 Core debug

Core debug is accessed through the core debug registers. Debug access to these registers is by means of the *Advanced High-performance Bus* (AHB-AP) port. The processor can access these registers directly over the internal *Private Peripheral Bus* (PPB).

It consists of 4 registers:

**Table 339. Core debug registers**

Register	Description
DHCSR	The 32-bit Debug Halting Control and Status Register: This provides status information about the state of the processor enable core debug halt and step the processor.
DCRSR	The 17-bit Debug Core Register Selector Register: This selects the processor register to transfer data to or from.
DCRDR	The 32-bit Debug Core Register Data Register: This holds data for reading and writing registers to and from the processor selected by the DCRSR (Selector) register.
DEMCR	The 32-bit Debug Exception and Monitor Control Register: This provides Vector Catching and Debug Monitor Control. This register contains a bit named <b>TRCENA</b> which enable the use of a TRACE.

*Note:* **Important:** these registers are not reset by a system reset. They are only reset by a power-on reset.

Refer to the *Cortex<sup>®</sup>-M4 r0p1 TRM* for further details.

To Halt on reset, it is necessary to:

- enable the bit0 (VC\_CORRESET) of the Debug and Exception Monitor Control Register
- enable the bit0 (C\_DEBUGEN) of the Debug Halting Control and Status Register.

## 51.11 Capability of the debugger host to connect under system reset

The STM32L4Rxxx and STM32L4Sxxx MCUs' reset system comprises the following reset sources:

- POR (power-on reset) which asserts a RESET at each power-up
- Internal watchdog reset
- Software reset
- External reset.

The Cortex<sup>®</sup>-M4 differentiates the reset of the debug part (generally PORRESETn) and the other one (SYSRESETn).

This way, it is possible for the debugger to connect under System Reset, programming the Core Debug Registers to halt the core when fetching the reset vector. Then the host can release the system reset and the core will immediately halt without having executed any instructions. In addition, it is possible to program any debug features under System Reset.

*Note:* It is highly recommended for the debugger host to connect (set a breakpoint in the reset vector) under system reset.

## 51.12 FPB (Flash patch breakpoint)

The FPB unit:

- implements hardware breakpoints
- patches code and data from code space to system space. This feature gives the possibility to correct software bugs located in the Code Memory Space.

The use of a Software Patch or a Hardware Breakpoint is exclusive.

The FPB consists of:

- 2 literal comparators for matching against literal loads from Code Space and remapping to a corresponding area in the System Space
- 6 instruction comparators for matching against instruction fetches from Code Space. They can be used either to remap to a corresponding area in the System Space or to generate a Breakpoint Instruction to the core.

## 51.13 DWT (data watchpoint trigger)

The DWT unit consists of four comparators. They are configurable as:

- a hardware watchpoint or
- a trigger to an ETM or
- a PC sampler or
- a data address sampler

The DWT also provides some means to give some profiling informations. For this, some counters are accessible to give the number of:

- Clock cycle
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- CPI (clock per instructions)
- Interrupt overhead



## 51.14 ITM (instrumentation trace macrocell)

### 51.14.1 General description

The ITM is an application-driven trace source that supports *printf* style debugging to trace *Operating System* (OS) and application events, and emits diagnostic system information. The ITM emits trace information as packets which can be generated as:

- **Software trace.** Software can write directly to the ITM stimulus registers to emit packets.
- **Hardware trace.** The DWT generates these packets, and the ITM emits them.
- **Time stamping.** Timestamps are emitted relative to packets. The ITM contains a 21-bit counter to generate the timestamp. The Cortex<sup>®</sup>-M4 clock or the bit clock rate of the *Serial Wire Viewer* (SWV) output clocks the counter.

The packets emitted by the ITM are output to the TPIU (Trace Port Interface Unit). The formatter of the TPIU adds some extra packets (refer to TPIU) and then output the complete packets sequence to the debugger host.

The bit TRCEN of the Debug Exception and Monitor Control Register must be enabled before you program or use the ITM.

### 51.14.2 Time stamp packets, synchronization and overflow packets

Time stamp packets encode time stamp information, generic control and synchronization. It uses a 21-bit timestamp counter (with possible prescalers) which is reset at each time stamp packet emission. This counter can be either clocked by the CPU clock or the SWV clock.

A synchronization packet consists of 6 bytes equal to 0x80\_00\_00\_00\_00\_00 which is emitted to the TPIU as 00 00 00 00 00 80 (LSB emitted first).

A synchronization packet is a timestamp packet control. It is emitted at each DWT trigger.

For this, the DWT must be configured to trigger the ITM: the bit CYCCNTENA (bit0) of the DWT Control Register must be set. In addition, the bit2 (SYNCENA) of the ITM Trace Control Register must be set.

*Note:* *If the SYNENA bit is not set, the DWT generates Synchronization triggers to the TPIU which will send only TPIU synchronization packets and not ITM synchronization packets.*

An overflow packet consists is a special timestamp packets which indicates that data has been written but the FIFO was full.

**Table 340. Main ITM registers**

Address	Register	Details
@E0000FB0	ITM lock access	Write 0xC5ACCE55 to unlock Write Access to the other ITM registers

**Table 340. Main ITM registers (continued)**

Address	Register	Details
@E0000E80	ITM trace control	Bits 31-24 = Always 0
		Bits 23 = Busy
		Bits 22-16 = 7-bits ATB ID which identifies the source of the trace data
		Bits 15-10 = Always 0
		Bits 9:8 = TSPrescale = Time Stamp Prescaler
		Bits 7-5 = Reserved
		Bit 4 = SWOENA = Enable SWV behavior (to clock the timestamp counter by the SWV clock)
		Bit 3 = DWTENA: Enable the DWT Stimulus
		Bit 2 = SYNCENA: this bit must be to 1 to enable the DWT to generate synchronization triggers so that the TPIU can then emit the synchronization packets
		Bit 1 = TSENA (Timestamp Enable)
Bit 0 = ITMENA: Global Enable Bit of the ITM		
@E0000E40	ITM trace privilege	Bit 3: mask to enable tracing ports31:24
		Bit 2: mask to enable tracing ports23:16
		Bit 1: mask to enable tracing ports15:8
		Bit 0: mask to enable tracing ports7:0
@E0000E00	ITM trace enable	Each bit enables the corresponding Stimulus port to generate trace
@E0000000- E000007C	Stimulus port registers 0-31	Write the 32-bits data on the selected Stimulus Port (32 available) to be traced out

**Example of configuration**

To output a simple value to the TPIU:

- Configure the TPIU and assign TRACE I/Os by configuring the DBGMCU\_CR (refer to [Section 51.17.2: TRACE pin assignment](#) and [Section 51.16.3: Debug MCU configuration register \(DBGMCU\\_CR\)](#))
- Write 0xC5ACCE55 to the ITM Lock Access Register to unlock the write access to the ITM registers
- Write 0x00010005 to the ITM Trace Control Register to enable the ITM with Synchronous enabled and an ATB ID different from 0x00
- Write 0x1 to the ITM Trace Enable Register to enable the Stimulus Port 0
- Write 0x1 to the ITM Trace Privilege Register to unmask Stimulus Ports 7:0
- Write the value to output in the Stimulus Port Register 0: this can be done by software (using a printf function)



## 51.15 ETM (Embedded trace macrocell)

### 51.15.1 General description

The ETM enables the reconstruction of program execution. Data are traced using the Data Watchpoint and Trace (DWT) component or the Instruction Trace Macrocell (ITM) whereas instructions are traced using the Embedded Trace Macrocell (ETM).

The ETM transmits information as packets and is triggered by embedded resources. These resources must be programmed independently and the trigger source is selected using the Trigger Event Register (0xE0041008). An event could be a simple event (address match from an address comparator) or a logic equation between 2 events. The trigger source is one of the four comparators of the DWT module, The following events can be monitored:

- Clock cycle matching
- Data address matching

For more informations on the trigger resources refer to [Section 51.13: DWT \(data watchpoint trigger\)](#).

The packets transmitted by the ETM are output to the TPIU (Trace Port Interface Unit). The formatter of the TPIU adds some extra packets (refer to [Section 51.17: TPIU \(trace port interface unit\)](#)) and then outputs the complete packet sequence to the debugger host.

### 51.15.2 Signal protocol, packet types

This part is described in the section 7 ETMv3 Signal Protocol of the Arm® IHI 0014N document.

### 51.15.3 Main ETM registers

For more information on registers refer to the chapter 3 of the Arm® IHI 0014N specification.

**Table 341. Main ETM registers**

Address	Register	Details
0xE0041FB0	ETM Lock Access	Write 0xC5ACCE55 to unlock the write access to the other ETM registers.
0xE0041000	ETM Control	This register controls the general operation of the ETM, for instance how tracing is enabled.
0xE0041010	ETM Status	This register provides information about the current status of the trace and trigger logic.
0xE0041008	ETM Trigger Event	This register defines the event that will control trigger.
0xE004101C	ETM Trace Enable Control	This register defines which comparator is selected.
0xE0041020	ETM Trace Enable Event	This register defines the trace enabling event.
0xE0041024	ETM Trace Start/Stop	This register defines the traces used by the trigger source to start and stop the trace, respectively.

### 51.15.4 Configuration example

To output a simple value to the TPIU:

- Configure the TPIU and enable the I/O\_TRACEN to assign TRACE I/Os in the STM32L4Rxxx and STM32L4Sxxx debug configuration register
- Write 0xC5ACCE55 to the ETM Lock Access Register to unlock the write access to the ITM registers
- Write 0x00001D1E to the control register (configure the trace)
- Write 0000406F to the Trigger Event register (define the trigger event)
- Write 0000006F to the Trace Enable Event register (define an event to start/stop)
- Write 00000001 to the Trace Start/stop register (enable the trace)
- Write 0000191E to the ETM Control Register (end of configuration)

## 51.16 MCU debug component (DBGMCU)

The MCU debug component helps the debugger provide support for:

- Low-power modes
- Clock control for timers, watchdog, I<sup>2</sup>C and bxCAN during a breakpoint
- Control of the trace pins assignment

### 51.16.1 Debug support for low-power modes

To enter low-power mode, the instruction WFI or WFE must be executed.

The MCU implements several low-power modes which can either deactivate the CPU clock or reduce the power of the CPU.

The core does not allow FCLK or HCLK to be turned off during a debug session. As these are required for the debugger connection, during a debug, they must remain active. The MCU integrates special means to allow the user to debug software in low-power modes.

For this, the debugger host must first set some debug configuration registers to change the low-power mode behavior:

- In Sleep mode, DBG\_SLEEP bit of DBGMCU\_CR register must be previously set by the debugger. This will feed HCLK with the same clock that is provided to FCLK (system clock previously configured by the software).
- In Stop mode, the bit DBG\_STOP must be previously set by the debugger. This will enable the internal RC oscillator clock to feed FCLK and HCLK in Stop mode.
- In Standby mode, the bit DBG\_STANDBY must be previously set by the debugger. This will keep the regulators on, and enable the internal RC oscillator clock to feed FCLK and HCLK in Standby mode. A system reset is generated internally so that exiting from Standby is identical than fetching from reset.

The DBGMCU\_CR register can be written by the debugger under system reset. If the debugger host does not support these features, it is still possible to write this register by software.

### 51.16.2 Debug support for timers, RTC, watchdog, bxCAN and I<sup>2</sup>C

During a breakpoint, it is necessary to choose how the counter of timers,RTC and watchdog should behave:

- They can continue to count inside a breakpoint. This is usually required when a PWM is controlling a motor, for example.
- They can stop to count inside a breakpoint. This is required for watchdog purposes.

For the bxCAN, the user can choose to block the update of the receive register during a breakpoint.

For the I<sup>2</sup>C, the user can choose to block the SMBUS timeout during a breakpoint.

The DBGMCU freeze registers can be written by the debugger under system reset. If the debugger host does not support these features, it is still possible to write these registers by software.

### 51.16.3 Debug MCU configuration register (DBGMCU\_CR)

Address: 0xE004 2004

Power-on reset: 0x0000 0000

System reset: not affected

Access: Only 32-bit access supported

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACE MODE [1:0]	TRACE IOEN	Res.	Res.	DBG_ STAND BY	DBG_ STOP	DBG_ SLEEP	
								rw	rw	rw			rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:5 **TRACE\_MODE[1:0] and TRACE\_IOEN**: Trace pin assignment control

- With TRACE\_IOEN=0:
  - TRACE\_MODE=xx: TRACE pins not assigned (default state)
- With TRACE\_IOEN=1:
  - TRACE\_MODE=00: TRACE pin assignment for Asynchronous Mode
  - TRACE\_MODE=01: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 1
  - TRACE\_MODE=10: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 2
  - TRACE\_MODE=11: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 4

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DBG\_STANDBY**: Debug Standby mode

0: (FCLK=Off, HCLK=Off) The whole digital part is unpowered.

From software point of view, exiting from Standby is identical than fetching reset vector (except a few status bit indicated that the MCU is resuming from Standby)

1: (FCLK=On, HCLK=On) In this case, the digital part is not unpowered and FCLK and HCLK are provided by the internal RC oscillator which remains active. In addition, the MCU generate a system reset during Standby mode so that exiting from Standby is identical than fetching from reset.

Bit 1 **DBG\_STOP**: Debug Stop mode

0: (FCLK=Off, HCLK=Off) In STOP mode, the clock controller disables all clocks (including HCLK and FCLK). When exiting from STOP mode, the clock configuration is identical to the one after RESET (CPU clocked by the 8 MHz internal RC oscillator (HSI16)). Consequently, the software must reprogram the clock controller to enable the PLL, the Xtal, etc.

1: (FCLK=On, HCLK=On) In this case, when entering STOP mode, FCLK and HCLK are provided by the internal RC oscillator which remains active in STOP mode. When exiting STOP mode, the software must reprogram the clock controller to enable the PLL, the Xtal, etc. (in the same way it would do in case of DBG\_STOP=0)

Bit 0 **DBG\_SLEEP**: Debug Sleep mode

0: (FCLK=On, HCLK=Off) In Sleep mode, FCLK is clocked by the system clock as previously configured by the software while HCLK is disabled.

In Sleep mode, the clock controller configuration is not reset and remains in the previously programmed state. Consequently, when exiting from Sleep mode, the software does not need to reconfigure the clock controller.

1: (FCLK=On, HCLK=On) In this case, when entering Sleep mode, HCLK is fed by the same clock that is provided to FCLK (system clock as previously configured by the software).

**51.16.4 Debug MCU APB1 freeze register1(DBGMCU\_APB1FZR1)**

Address: 0xE004 2008

Power on reset (POR): 0x0000 0000

System reset: not affected

Access: Only 32-bit access are supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1_STOP	Res.	Res.	Res.	Res.	Res.	DBG_CAN1_STOP	Res.	DBG_I2C3_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	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.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Res.	Res.	Res.	Res.	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
			rw	rw	rw					rw	rw	rw	rw	rw	rw

- Bit 31 **DBG\_LPTIM1\_STOP**: LPTIM1 counter stopped when core is halted  
0: The counter clock of LPTIM1 is fed even if the core is halted  
1: The counter clock of LPTIM1 is stopped when the core is halted
- Bits 30:26 Reserved, must be kept at reset value.
- Bit 25 **DBG\_CAN1\_STOP**: bxCAN1 stopped when core is halted  
0: Same behavior as in normal mode  
1: The bxCAN1 receive registers are frozen
- Bit 24 Reserved, must be kept at reset value.
- Bit 23 **DBG\_I2C3\_STOP**: I2C3 SMBUS timeout counter stopped when core is halted  
0: Same behavior as in normal mode  
1: The I2C3 SMBus timeout is frozen
- Bit 22 **DBG\_I2C2\_STOP**: I2C2 SMBUS timeout counter stopped when core is halted  
0: Same behavior as in normal mode  
1: The I2C2 SMBus timeout is frozen
- Bit 21 **DBG\_I2C1\_STOP**: I2C1 SMBUS timeout counter stopped when core is halted  
0: Same behavior as in normal mode  
1: The I2C1 SMBus timeout is frozen
- Bits 20:13 Reserved, must be kept at reset value.
- Bit 12 **DBG\_IWDG\_STOP**: Independent watchdog counter stopped when core is halted  
0: The independent watchdog counter clock continues even if the core is halted  
1: The independent watchdog counter clock is stopped when the core is halted
- Bit 11 **DBG\_WWDG\_STOP**: Window watchdog counter stopped when core is halted  
0: The window watchdog counter clock continues even if the core is halted  
1: The window watchdog counter clock is stopped when the core is halted
- Bit 10 **DBG\_RTC\_STOP**: RTC counter stopped when core is halted  
0: The clock of the RTC counter is fed even if the core is halted  
1: The clock of the RTC counter is stopped when the core is halted
- Bits 9:6 Reserved, must be kept at reset value.
- Bit 5 **DBG\_TIM7\_STOP**: TIM7 counter stopped when core is halted  
0: The counter clock of TIM7 is fed even if the core is halted  
1: The counter clock of TIM7 is stopped when the core is halted
- Bit 4 **DBG\_TIM6\_STOP**: TIM6 counter stopped when core is halted  
0: The counter clock of TIM6 is fed even if the core is halted  
1: The counter clock of TIM6 is stopped when the core is halted
- Bit 3 **DBG\_TIM5\_STOP**: TIM5 counter stopped when core is halted  
0: The counter clock of TIM5 is fed even if the core is halted  
1: The counter clock of TIM5 is stopped when the core is halted

- Bit 2 **DBG\_TIM4\_STOP**: TIM4 counter stopped when core is halted
  - 0: The counter clock of TIM4 is fed even if the core is halted
  - 1: The counter clock of TIM4 is stopped when the core is halted
- Bit 1 **DBG\_TIM3\_STOP**: TIM3 counter stopped when core is halted
  - 0: The counter clock of TIM3 is fed even if the core is halted
  - 1: The counter clock of TIM3 is stopped when the core is halted
- Bit 0 **DBG\_TIM2\_STOP**: TIM2 counter stopped when core is halted
  - 0: The counter clock of TIM2 is fed even if the core is halted
  - 1: The counter clock of TIM2 is stopped when the core is halted

### 51.16.5 Debug MCU APB1 freeze register 2 (DBGMCU\_APB1FZR2)

Address: 0xE004 200C

Power on reset (POR): 0x0000 0000

System reset: not affected

Access: Only 32-bit access are supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM2_STOP	Res.	Res.	Res.	DBG_I2C4_STOP	Res.
										rw				rw	

Bits 31:6 Reserved, must be kept at reset value.

- Bit 5 **DBG\_LPTIM2\_STOP**: LPTIM2 counter stopped when core is halted
  - 0: The counter clock of LPTIM2 is fed even if the core is halted
  - 1: The counter clock of LPTIM2 is stopped when the core is halted

Bits 4:2 Reserved, must be kept at reset value

- Bit 1 **DBG\_I2C4\_STOP**: I2C4 SMBUS timeout counter stopped when core is halted
  - 0: Same behavior as in normal mode
  - 1: The I2C4 SMBus timeout is frozen

Bit 0 Reserved, must be kept at reset value

### 51.16.6 Debug MCU APB2 freeze register (DBGMCU\_APB2FZR)

Address: 0xE004 2010

Power on reset (POR): 0x0000 0000

System reset: not affected

Access: Only 32-bit access are supported.



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DBG_TIM8_STOP	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rw		rw											

Bits 31:19 Reserved, must be kept at reset value.

- Bit 18 **DBG\_TIM17\_STOP**: TIM17 counter stopped when core is halted
  - 0: The clock of the TIM17 counter is fed even if the core is halted
  - 1: The clock of the TIM17 counter is stopped when the core is halted
- Bit 17 **DBG\_TIM16\_STOP**: TIM16 counter stopped when core is halted
  - 0: The clock of the TIM16 counter is fed even if the core is halted
  - 1: The clock of the TIM16 counter is stopped when the core is halted
- Bit 16 **DBG\_TIM15\_STOP**: TIM15 counter stopped when core is halted
  - 0: The clock of the TIM15 counter is fed even if the core is halted
  - 1: The clock of the TIM15 counter is stopped when the core is halted

Bits 15:14 Reserved, must be kept at reset value.

- Bit 13 **DBG\_TIM8\_STOP**: TIM8 counter stopped when core is halted
  - 0: The clock of the TIM8 counter is fed even if the core is halted
  - 1: The clock of the TIM8 counter is stopped when the core is halted
- Bit 12 Reserved, must be kept at reset value.
- Bit 11 **DBG\_TIM1\_STOP**: TIM1 counter stopped when core is halted
  - 0: The clock of the TIM1 counter is fed even if the core is halted
  - 1: The clock of the TIM1 counter is stopped when the core is halted

Bits 10:0 Reserved, must be kept at reset value.

## 51.17 TPIU (trace port interface unit)

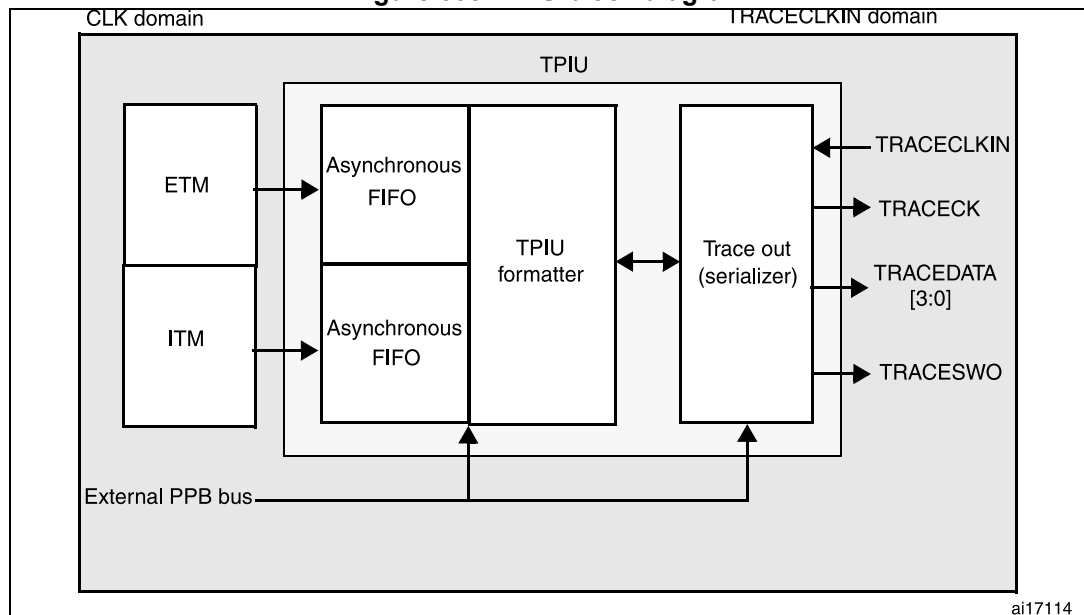
### 51.17.1 Introduction

The TPIU acts as a bridge between the on-chip trace data from the ITM and the ETM.

The output data stream encapsulates the trace source ID, that is then captured by a *trace port analyzer* (TPA).

The core embeds a simple TPIU, especially designed for low-cost debug (consisting of a special version of the CoreSight TPIU).

Figure 608. TPIU block diagram



### 51.17.2 TRACE pin assignment

- Asynchronous mode  
The asynchronous mode requires 1 extra pin and is available on all packages. It is only available if using Serial Wire mode (not in JTAG mode).

Table 342. Asynchronous TRACE pin assignment

TPUI pin name	Trace synchronous mode		STM32L4Rxxx and STM32L4Sxxx pin assignment
	Type	Description	
TRACESWO	O	TRACE Asynchronous Data Output	PB3

- Synchronous mode  
The synchronous mode requires from 2 to 6 extra pins depending on the data trace size and is only available in the larger packages. In addition it is available in JTAG mode and in Serial Wire mode and provides better bandwidth output capabilities than asynchronous trace.

**Table 343. Synchronous TRACE pin assignment**

TPUI pin name	Trace synchronous mode		STM32L4Rxxx and STM32L4Sxxx pin assignment
	Type	Description	
TRACECK	O	TRACE Clock	PE2
TRACED[3:0]	O	TRACE Synchronous Data Outputs Can be 1, 2 or 4.	PE[6:3] PC1, PC8, PD2, PC12

**TPUI TRACE pin assignment**

By default, these pins are NOT assigned. They can be assigned by setting the TRACE\_IOEN and TRACE\_MODE bits in the *Debug MCU configuration register (DBGMCU\_CR)*. This configuration has to be done by the debugger host.

In addition, the number of pins to assign depends on the trace configuration (asynchronous or synchronous).

- **Asynchronous mode:** 1 extra pin is needed
- **Synchronous mode:** from 2 to 5 extra pins are needed depending on the size of the data trace port register (1, 2 or 4) :
  - TRACECK
  - TRACED(0) if port size is configured to 1, 2 or 4
  - TRACED(1) if port size is configured to 2 or 4
  - TRACED(2) if port size is configured to 4
  - TRACED(3) if port size is configured to 4

To assign the TRACE pin, the debugger host must program the bits TRACE\_IOEN and TRACE\_MODE[1:0] of the *Debug MCU configuration register (DBGMCU\_CR)*. By default the TRACE pins are not assigned.

This register is mapped on the external PPB and is reset by the PORESET (and not by the SYSTEM reset). It can be written by the debugger under SYSTEM reset.

**Table 344. Flexible TRACE pin assignment**

DBGMCU_CR register		Pins assigned for:	TRACE IO pin assigned				
TRACE_IOEN	TRACE_MODE [1:0]		PB3 / JTDO / TRACESWO	PE2 / TRACECK	PE3 / TRACED[0]	PE4 / TRACED[1]	PE5 / TRACED[2]
0	XX	No Trace (default state)	Released <sup>(1)</sup>	-			
1	00	Asynchronous Trace	TRACESWO	-	-	Released (usable as GPIO)	

**Table 344. Flexible TRACE pin assignment (continued)**

DBGMCU_CR register		Pins assigned for:	TRACE IO pin assigned					
TRACE_IOEN	TRACE_MODE [1:0]		PB3 / JTDO / TRACESWO	PE2 / TRACECK	PE3 / TRACED[0]	PE4 / TRACED[1]	PE5 / TRACED[2]	PE6 / TRACED[3]
1	01	Synchronous Trace 1 bit	Released <sup>(1)</sup>	TRACECK	TRACED[0]	-	-	-
1	10	Synchronous Trace 2 bit		TRACECK	TRACED[0]	TRACED[1]	-	-
1	11	Synchronous Trace 4 bit		TRACECK	TRACED[0]	TRACED[1]	TRACED[2]	TRACED[3]

1. When Serial Wire mode is used, it is released, but when JTAG is used, it is assigned to JTDO.

*Note:* By default, the TRACECLKIN input clock of the TPIU is tied to GND. It is assigned to HCLK two clock cycles after the bit TRACE\_IOEN has been set.

The debugger must then program the Trace Mode by writing the PROTOCOL[1:0] bits in the SPP\_R (Selected Pin Protocol) register of the TPIU.

- PROTOCOL=00: Trace Port Mode (synchronous)
- PROTOCOL=01 or 10: Serial Wire (Manchester or NRZ) Mode (asynchronous mode). Default state is 01

It then also configures the TRACE port size by writing the bits [3:0] in the CPSPS\_R (Current Synchronous Port Size Register) of the TPIU:

- 0x1 for 1 pin (default state)
- 0x2 for 2 pins
- 0x8 for 4 pins

### 51.17.3 TPUI formatter

The formatter protocol outputs data in 16-byte frames:

- seven bytes of data
- eight bytes of mixed-use bytes consisting of:
  - 1 bit (LSB) to indicate it is a DATA byte ('0) or an ID byte ('1).
  - 7 bits (MSB) which can be data or change of source ID trace.
- one byte of auxiliary bits where each bit corresponds to one of the eight mixed-use bytes:
  - if the corresponding byte was a data, this bit gives bit0 of the data.
  - if the corresponding byte was an ID change, this bit indicates when that ID change takes effect.

*Note:* Refer to the Arm® CoreSight Architecture Specification v1.0 (Arm IHI 0029B) for further information

### 51.17.4 TPUI frame synchronization packets

The TPUI can generate two types of synchronization packets:

- The Frame Synchronization packet (or Full Word Synchronization packet)  
It consists of the word: 0x7F\_FF\_FF\_FF (LSB emitted first). This sequence can not occur at any other time provided that the ID source code 0x7F has not been used.  
It is output periodically **between** frames.  
In continuous mode, the TPA must discard all these frames once a synchronization frame has been found.
- The Half-Word Synchronization packet  
It consists of the half word: 0x7F\_FF (LSB emitted first).  
It is output periodically **between or within** frames.  
These packets are only generated in continuous mode and enable the TPA to detect that the TRACE port is in IDLE mode (no TRACE to be captured). When detected by the TPA, it must be discarded.

### 51.17.5 Transmission of the synchronization frame packet

There is no Synchronization Counter register implemented in the TPIU of the core. Consequently, the synchronization trigger can only be generated by the **DWT**. Refer to the registers DWT Control Register (bits SYNCTAP[11:10]) and the DWT Current PC Sampler Cycle Count Register.

The TPUI Frame synchronization packet (0x7F\_FF\_FF\_FF) is emitted:

- after each TPIU reset release. This reset is synchronously released with the rising edge of the TRACECLKIN clock. This means that this packet is transmitted when the TRACE\_IOEN bit in the DBGMCU\_CFG register is set. In this case, the word 0x7F\_FF\_FF\_FF is not followed by any formatted packet.
- at each DWT trigger (assuming DWT has been previously configured). Two cases occur:
  - If the bit SYNENA of the ITM is reset, only the word 0x7F\_FF\_FF\_FF is emitted without any formatted stream which follows.
  - If the bit SYNENA of the ITM is set, then the ITM synchronization packets will follow (0x80\_00\_00\_00\_00\_00), formatted by the TPUI (trace source ID added).

### 51.17.6 Synchronous mode

The trace data output size can be configured to 4, 2 or 1 pin: TRACED(3:0)

The output clock is output to the debugger (TRACECK)

Here, TRACECLKIN is driven internally and is connected to HCLK only when TRACE is used.

*Note:* In this synchronous mode, it is not required to provide a stable clock frequency.

The TRACE I/Os (including TRACECK) are driven by the rising edge of TRACLKIN (equal to HCLK). Consequently, the output frequency of TRACECK is equal to HCLK/2.

### 51.17.7 Asynchronous mode

This is a low cost alternative to output the trace using only 1 pin: this is the asynchronous output pin TRACESWO. Obviously there is a limited bandwidth.

TRACESWO is multiplexed with JTDO when using the SW-DP pin. This way, this functionality is available in all STM32L4Rxxx and STM32L4Sxxx packages.

This asynchronous mode requires a constant frequency for TRACECLKIN. For the standard UART (NRZ) capture mechanism, 5% accuracy is needed. The Manchester encoded version is tolerant up to 10%.

### 51.17.8 TRACECLKIN connection inside the STM32L4Rxxx and STM32L4Sxxx

In the STM32L4Rxxx and STM32L4Sxxx, this TRACECLKIN input is internally connected to HCLK. This means that when in asynchronous trace mode, the application is restricted to use time frames where the CPU frequency is stable.

*Note:* **Important:** when using asynchronous trace: it is important to be aware that:

*The default clock of the STM32L4Rxxx and STM32L4Sxxx MCUs is the internal RC oscillator. Its frequency under reset is different from the one after reset release. This is because the RC calibration is the default one under system reset and is updated at each system reset release.*

*Consequently, the trace port analyzer (TPA) should not enable the trace (with the TRACE\_IOEN bit) under system reset, because a Synchronization Frame Packet will be issued with a different bit time than trace packets which will be transmitted after reset release.*

### 51.17.9 TPIU registers

The TPIU APB registers can be read and written only if the bit TRCENA of the Debug Exception and Monitor Control Register (DEMCR) is set. Otherwise, the registers are read as zero (the output of this bit enables the PCLK of the TPIU).

**Table 345. Important TPIU registers**

Address	Register	Description
0xE0040004	Current port size	Allows the trace port size to be selected: Bit 0: Port size = 1 Bit 1: Port size = 2 Bit 2: Port size = 3, not supported Bit 3: Port Size = 4  Only 1 bit must be set. By default, the port size is one bit. (0x00000001)
0xE00400F0	Selected pin protocol	Allows the Trace Port Protocol to be selected: Bit1:0 = 00: Synchronous Trace Port Mode 01: Serial Wire Output - manchester (default value) 10: Serial Wire Output - NRZ 11: reserved
0xE0040304	Formatter and flush control	Bit 31-9 = always '0' Bit 8 = TriglIn = always '1' to indicate that triggers are indicated Bit 7-4 = always 0 Bit 3-2 = always 0 Bit 1 = EnFCont. In Synchronous Trace mode (Select_Pin_Protocol register bit1:0 = 00), this bit is forced to '1': the formatter is automatically enabled in continuous mode. In asynchronous mode (Select_Pin_Protocol register bit1:0 <> 00), this bit can be written to activate or not the formatter. Bit 0 = always '0'  The resulting default value is 0x102 <b>Note:</b> In synchronous mode, because the TRACECTL pin is not mapped outside the chip, the formatter is always enabled in continuous mode; this way the formatter inserts some control packets to identify the source of the trace packets).
0xE0040300	Formatter and flush status	Not used in Cortex <sup>®</sup> -M4, always read as 0x00000008

### 51.17.10 Example of configuration

- Set the bit TRCENA in the Debug Exception and Monitor Control Register (DEMCR)
- Write the TPIU Current Port Size Register to the desired value (default is 0x1 for a 1-bit port size)
- Write TPIU Formatter and Flush Control Register to 0x102 (default value)
- Write the TPIU Select Pin Protocol to select the synchronous or asynchronous mode. Example: 0x2 for asynchronous NRZ mode (UART like)
- Write the DBGMCU control register to 0x20 (bit IO\_TRACEN) to assign TRACE I/Os for asynchronous mode. A TPIU Synchronous packet is emitted at this time (FF\_FF\_FF\_7F)
- Configure the ITM and write the ITM Stimulus register to output a value



## 51.18 DBG register map

The following table summarizes the Debug registers

**Table 346. DBG register map and reset values**

Addr.	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE0042000	DBGMCU_IDCODE	REV_ID												DEV_ID																			
	Reset value <sup>(1)</sup>	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X						X	X	X	X	X	X	X	X	X	X	X
0xE0042004	DBGMCU_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																										0	0	0	0	0	0	0
0xE004 2008	DBGMCU_APB1FZR1	DBG_LPTIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	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
0xE004 200C	DBGMCU_APB1FZR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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				
0xE004 2010	DBGMCU_APB2FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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										

1. The reset value is product dependent. For more information, refer to [Section 51.6.1: MCU device ID code](#).

## 52 Device electronic signature

The device electronic signature is stored in the System memory area of the Flash memory module, and can be read using the debug interface or by the CPU. It contains factory-programmed identification and calibration data that allow the user firmware or other external devices to automatically match to the characteristics of the STM32L4Rxxx and STM32L4Sxxx microcontroller.

### 52.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

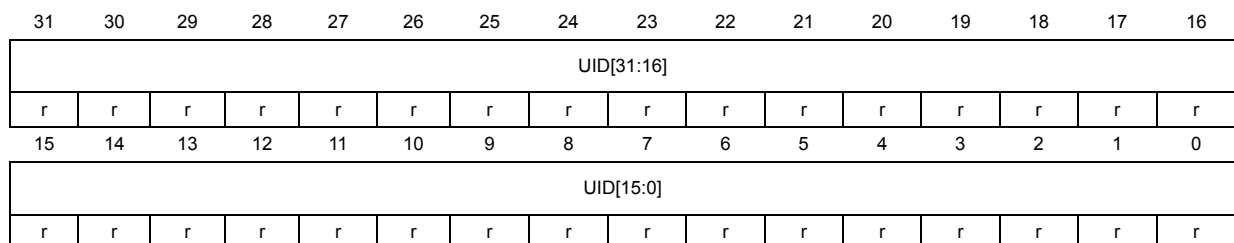
- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as part of the security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory
- to activate secure boot processes, etc.

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits cannot be altered by the user.

Base address: 0x1FFF 7590

Address offset: 0x00

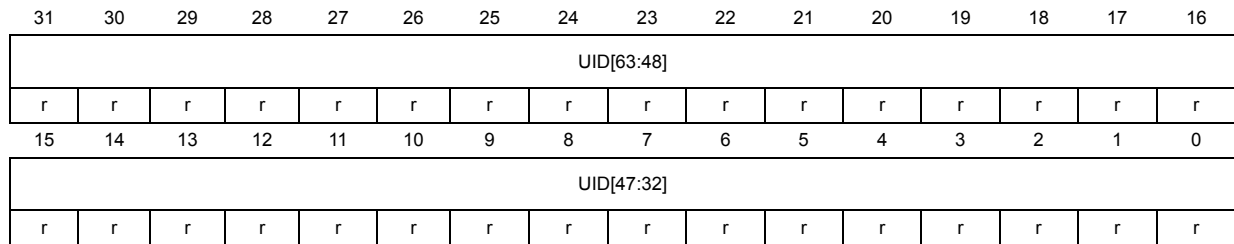
Read only = 0xXXXX XXXX where X is factory-programmed



Bits 31:0 **UID[31:0]**: X and Y coordinates on the wafer

Address offset: 0x04

Read only = 0xXXXX XXXX where X is factory-programmed

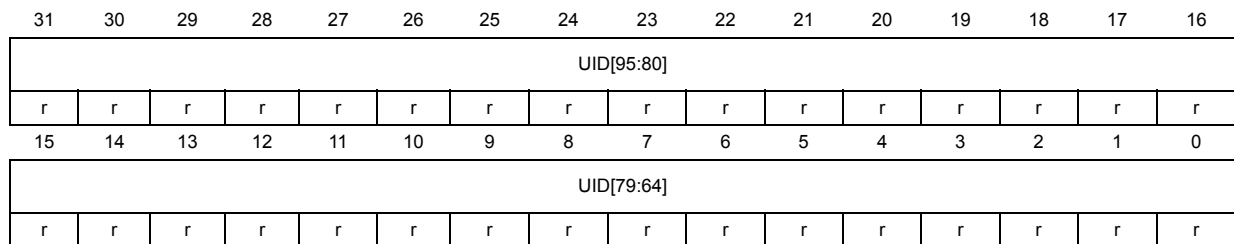


Bits 31:8 **UID[63:40]:** LOT\_NUM[23:0]  
 Lot number (ASCII encoded)

Bits 7:0 **UID[39:32]:** WAF\_NUM[7:0]  
 Wafer number (8-bit unsigned number)

Address offset: 0x08

Read only = 0xXXXX XXXX where X is factory-programmed



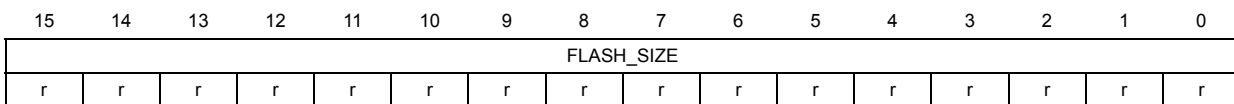
Bits 31:0 **UID[95:64]:** LOT\_NUM[55:24]  
 Lot number (ASCII encoded)

## 52.2 Flash size data register

Base address: 0x1FFF 75E0

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed



Bits 15:0 **FLASH\_SIZE[15:0]:** Flash memory size  
 This bitfield indicates the size of the device Flash memory expressed in Kbytes.  
 As an example, 0x040 corresponds to 64 Kbytes.

### 52.3 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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

00010: LQFP100 without DSI

00011: UFBGA132 without DSI

001XX: LQFP144 without DSI

10000: UFBGA169 without DSI

10010: LQFP100 with DSI

10011: UFBGA144 with DSI, WLCSP144 with DSI

10100: UFBGA169 with DSI

10101: LQFP144 with DSI

Others: reserved

# 53 Revision history

**Table 347. Document revision history**

Date	Revision	Changes
13-Oct-2017	1	Initial release.
05-Feb-2018	2	<p>Added:</p> <ul style="list-style-type: none"> <li>– <a href="#">Section 3.7.17: Flash configuration register (FLASH_CFGR)</a></li> <li>– <a href="#">Section 12.6.7: DMAMUX size identification register (DMAMUX_SIDR)</a></li> <li>– <a href="#">Section 12.6.8: DMAMUX IP identification register (DMAMUX_IPIDR)</a></li> <li>– <a href="#">Section 12.6.9: DMAMUX version register (DMAMUX_VERR)</a></li> <li>– <a href="#">Section 12.6.10: DMAMUX hardware configuration 1 register (DMAMUX_HWCFGR1)</a></li> <li>– <a href="#">Section 12.6.11: DMAMUX hardware configuration 2 register (DMAMUX_HWCFGR2)</a></li> <li>– <a href="#">CS boundary, refresh and communication regulation features on Section 19.2.4: Common functionality between the regular-command mode and HyperBus™</a></li> <li>– <a href="#">Status flag polling mode configuration on Section 19.4.3: OCTOSPI regular-command mode configuration</a></li> <li>– <a href="#">Section 19.4.7: OCTOSPI reconfiguration or deactivation</a></li> <li>– <a href="#">Section 19.6.5: OCTOSPI device configuration register 4 (OCTOSPI_DCR4)</a></li> <li>– <a href="#">Section 38.4.13: Timer counter reset</a></li> <li>– <a href="#">Section 50.4.2: USB OTG pin and internal signals</a></li> <li>– <a href="#">Table 321: Compatibility of STM32 low power modes with the OTG</a></li> <li>– <a href="#">Figure 585: Updating OTG_HFIR dynamically (RLDCTRL = 0)</a></li> </ul> <p>Deleted:</p> <ul style="list-style-type: none"> <li>– Figure CS high time and clocked CS high timer</li> <li>– OCTOSPI registers: OCTOSPI HW configuration register, (OCTOSPI_HWCFGR), OCTOSPI version register (OCTOSPI_VER), OCTOSPI identification (OCTOSPI_ID), OCTOSPI HW magic ID (OCTOSPI_MID)</li> </ul>

**Table 347. Document revision history (continued)**

Date	Revision	Changes
05-Feb-2018	2 Continued	<p>Updated (Continued):</p> <ul style="list-style-type: none"> <li>– <a href="#">Section 1.2: List of abbreviations for registers</a></li> <li>– <a href="#">Section 2.4: Embedded SRAM</a></li> <li>– <a href="#">Section 2.6.1: Boot configuration</a></li> <li>– <a href="#">Section 3.3.2: Error code correction (ECC)</a></li> <li>– <a href="#">Table 17: Flash interface - register map and reset values</a></li> <li>– <a href="#">Section 5.1: Power supplies</a></li> <li>– <a href="#">Figure 7: STM32L4S5xx/R5xx and STM32L4S7xx/L4R7xx power supply overview</a></li> <li>– <a href="#">Figure 8: STM32L4S9xx/L4R9xx power supply overview</a></li> <li>– <a href="#">Section 5.3.9: Standby mode</a></li> <li>– <a href="#">Section 5.3.10: Shutdown mode</a></li> <li>– <a href="#">Section 5.3.9: Standby mode</a></li> <li>– <a href="#">Section 6.2: Clocks</a></li> <li>– <a href="#">Section 6.2.11: Clock security system on LSE</a></li> <li>– <a href="#">Section 6.4.4: PLL configuration register (RCC_PLLCFGR)</a></li> <li>– <a href="#">Section 8.3.8: External interrupt/wakeup lines</a></li> <li>– <a href="#">Section 8.4.5: GPIO port input data register (GPIOx_IDR) (x = A to I)</a></li> <li>– <a href="#">Table 43: STM32L4Rxxx and STM32L4Sxxx peripherals interconnect matrix</a></li> <li>– <a href="#">Section 10.3.3: From ADC to timer (TIM1/TIM8)</a></li> <li>– <a href="#">Section 10.3.5: From timer (TIM1/TIM3/TIM4/TIM6/TIM7/TIM8/TIM16/LPTIM1/LPTIM2) and EXTI to DFSDM1</a></li> <li>– <a href="#">Section Table 50.: DMAMUX: assignment of multiplexer inputs to resources</a></li> <li>– <a href="#">Section 12.4.4: DMAMUX request line multiplexer</a></li> <li>– <a href="#">Section 12.4.5: DMAMUX request generator</a></li> <li>– <a href="#">Section 12.6.1: DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR)</a></li> <li>– <a href="#">Section 19.6.7: DMAMUX1 request generator channel x configuration register (DMAMUX1_RGxCR)</a></li> <li>– <a href="#">Figure 37: Virtual buffer and physical buffer memory map</a></li> <li>– <a href="#">Dummy-cycles phase on Section 19.2.1: OCTOSPI regular-command mode</a></li> <li>– <a href="#">Section 19.3.1: OCTOSPI indirect mode</a></li> <li>– <a href="#">Section 19.3.2: OCTOSPI status flag polling mode</a></li> <li>– <a href="#">Section 19.3.3: OCTOSPI memory-mapped mode</a></li> <li>– <a href="#">Section 19.4.1: OCTOSPI system configuration</a></li> <li>– <a href="#">Section 19.4.2: OCTOSPI device configuration</a></li> <li>– <a href="#">Memory-mapped mode configuration on Section 19.4.3: OCTOSPI regular-command mode configuration</a></li> <li>– <a href="#">Section 19.4.6: OCTOSPI busy bit and abort functionality</a></li> <li>– <a href="#">Section 19.6.1: OCTOSPI control register (OCTOSPI_CR)</a></li> <li>– <a href="#">Section 19.6.2: OCTOSPI device configuration register 1 (OCTOSPI_DCR1)</a></li> </ul>

**Table 347. Document revision history (continued)**

Date	Revision	Changes
05-Feb-2018	2 Continued	<p>Updated (Continued):</p> <ul style="list-style-type: none"> <li>– <a href="#">Section 19.6.3: OCTOSPI device configuration register 2 (OCTOSPI_DCR2)</a></li> <li>– <a href="#">Section 19.6.4: OCTOSPI device configuration register 3 (OCTOSPI_DCR3)</a></li> <li>– <a href="#">Section 19.6.13: OCTOSPI communication configuration register (OCTOSPI_CCR)</a></li> <li>– <a href="#">Section 19.6.14: OCTOSPI timing configuration register (OCTOSPI_TCR)</a></li> <li>– <a href="#">Section 19.6.19: OCTOSPI wrap communication configuration register (OCTOSPI_WPCCR)</a></li> <li>– <a href="#">Section 19.6.20: OCTOSPI wrap timing configuration register (OCTOSPI_WPTCR)</a></li> <li>– <a href="#">Section 19.6.21: OCTOSPI wrap instruction register (OCTOSPI_WPIR)</a></li> <li>– <a href="#">Section 19.6.22: OCTOSPI wrap alternate bytes register (OCTOSPI_WPABR)</a></li> <li>– <a href="#">Section 19.6.19: OCTOSPI write timing configuration register (OCTOSPI_WTCR)</a></li> <li>– <a href="#">Section 19.6.20: OCTOSPI write instruction register (OCTOSPI_WIR)</a></li> <li>– <a href="#">Section 19.6.21: OCTOSPI write alternate bytes register (OCTOSPI_WABR)</a></li> <li>– <a href="#">Section 19.6.22: OCTOSPI HyperBus™ latency configuration register (OCTOSPI_HLCR)</a></li> <li>– <a href="#">Section 19.6.23: OCTOSPI register map</a></li> <li>– <a href="#">Table 114: ADC internal input/output signals</a></li> <li>– <a href="#">Table 115: ADC input/output pins</a></li> <li>– <a href="#">Figure 86: ADC1 connectivity</a></li> <li>– <a href="#">Section 21.4.7: Single-ended and differential input channels</a></li> <li>– <a href="#">Table 121: Offset computation versus data resolution</a></li> <li>– <a href="#">Section 21.6: ADC registers</a> (all registers naming updated, no updates on the content of the registers unless otherwise specified)</li> <li>– <a href="#">Section 21.4.29: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)</a></li> <li>– <a href="#">Section 22.7.15: DAC calibration control register (DAC_CCR)</a></li> <li>– <a href="#">Section 23.7: DCMI register description</a> formatting of the reset values of all registers on this section were updated</li> <li>– <a href="#">Table 165: DFSDM break connection</a></li> <li>– <a href="#">Channel inputs selection on Section 27.4.4: Serial channel transceivers</a></li> <li>– <a href="#">Section 27.7: DFSDM channel y registers (y=0..7)</a> (all registers naming, no updates on the registers content unless otherwise specified)</li> <li>– <a href="#">Section 27.8.13: DFSDM filter x extremes detector maximum register (DFSDM_FLTxEXMAX)</a></li> </ul>

**Table 347. Document revision history (continued)**

Date	Revision	Changes
05-Feb-2018	2 Continued	<p>Updated (Continued):</p> <ul style="list-style-type: none"> <li>– <a href="#">Section 28.6: LTDC programming procedure</a></li> <li>– <a href="#">Section 28.7.21: LTDC layer x blending factors configuration register (LTDC_LxBFCR)</a></li> <li>– <a href="#">Section 30.6.10: TSC I/O group x counter register (TSC_IOGxCR)</a></li> <li>– <a href="#">Section 34.4.23: TIM1/TIM8 capture/compare mode register 3 (TIMx_CCMR3)</a></li> <li>– <a href="#">Figure 342: Capture/Compare channel 1 main circuit</a></li> <li>– <a href="#">Figure 343: Output stage of Capture/Compare channel (channel 1)</a></li> <li>– <a href="#">Figure 409: Low-power timer block diagram</a></li> <li>– <a href="#">Section 38.7.5: LPTIM control register (LPTIM_CR)</a></li> <li>– <a href="#">Section 40.4.4: Status register (IWDG_SR)</a></li> <li>– <a href="#">Figure 418: Watchdog block diagram</a></li> <li>– <a href="#">Figure 419: Window watchdog timing diagram</a></li> <li>– <a href="#">Section 42.3.11: RTC reference clock detection</a></li> <li>– <a href="#">Section 42.6.3: RTC control register (RTC_CR)</a></li> <li>– <a href="#">Section 43: Inter-integrated circuit (I2C) interface</a>: all section updated the SMBus version reference</li> <li>– <a href="#">Section 43.2: I2C main features</a></li> <li>– <a href="#">Table 248: I2C implementation</a></li> <li>– <a href="#">Figure 421: I2C block diagram</a></li> <li>– <a href="#">Figure 423: Setup and hold timings</a></li> <li>– <a href="#">Section 43.4.1: I2C block diagram</a></li> <li>– <a href="#">Section 43.6: I2C interrupts</a></li> <li>– <a href="#">Table 266: Tolerance of the USART receiver when BRR [3:0] = 0000</a></li> <li>– <a href="#">Table 267: Tolerance of the USART receiver when BRR[3:0] is different from 0000</a></li> <li>– <a href="#">Section 44.7: USART registers</a> in all subsections, the notes related to reserved bit/bitfield were updated</li> <li>– <a href="#">Section 45.5: LPUART registers</a> in all subsections, the notes related to reserved bit/bitfield were updated</li> <li>– <a href="#">Clock generator programming in SPDIF generator mode on Section 47.3.12: SPDIF output</a></li> <li>– <a href="#">Section 47.5.19: PDM delay register (SAI_PDMDLY)</a></li> <li>– <a href="#">Figure 564: CAN network topology</a></li> <li>– <a href="#">Figure 567: bxCAN in silent mode</a></li> <li>– <a href="#">Figure 570: Transmit mailbox states</a></li> <li>– <a href="#">Figure 572: Filter bank scale configuration - register organization</a></li> <li>– <a href="#">Figure 573: Example of filter numbering</a></li> <li>– <a href="#">Figure 576: Bit timing</a></li> <li>– <a href="#">Figure 574: Filtering mechanism - example</a></li> <li>– <a href="#">Figure 578: Event flags and interrupt generation</a></li> <li>– <a href="#">Figure 579: CAN mailbox registers</a></li> <li>– <a href="#">Section 50.1: Introduction</a></li> <li>– <a href="#">Table 318: OTG implementation</a></li> </ul>



**Table 347. Document revision history (continued)**

Date	Revision	Changes
05-Feb-2018	2 Continued	Updated (Continued): <ul style="list-style-type: none"> <li>– <a href="#">Figure 580: OTG full-speed block diagram</a></li> <li>– <a href="#">Section 50.7: USB host</a></li> <li>– <a href="#">Section 50.9: OTG low-power modes</a></li> <li>– <a href="#">Figure 588: Interrupt hierarchy</a></li> <li>– <a href="#">Section 50.15.1: OTG control and status register (OTG_GOTGCTL)</a></li> <li>– <a href="#">Section 50.15.4: OTG USB configuration register (OTG_GUSBCFG)</a></li> <li>– <a href="#">Section 50.15.7: OTG interrupt mask register (OTG_GINTMSK)</a></li> <li>– <a href="#">Section 50.15.8: OTG receive status debug read/OTG status read and pop registers (OTG_GRXSTSR/OTG_GRXSTSP)</a></li> <li>– <a href="#">Section 50.15.18: OTG device IN endpoint transmit FIFO size register (OTG_DIEPTxFx) (x = 1..5, where x is the FIFO number)</a></li> <li>– <a href="#">Section 50.15.21: OTG host frame interval register (OTG_HFIR)</a></li> <li>– <a href="#">Section 50.15.27: OTG host channel x characteristics register (OTG_HCCHARx) (x = 0..11, where x = Channel number)</a></li> <li>– <a href="#">Section 50.15.28: OTG host channel x interrupt register (OTG_HCINTx) (x = 0..11, where x = Channel number)</a></li> <li>– <a href="#">Section 50.15.29: OTG host channel x interrupt mask register (OTG_HCINTMSKx) (x = 0..11, where x = Channel number)</a></li> <li>– <a href="#">Section 50.15.32: OTG device configuration register (OTG_DCFG)</a></li> <li>– <a href="#">Section 50.15.35: OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)</a></li> <li>– <a href="#">Section 50.15.35: OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)</a></li> <li>– <a href="#">Section 50.15.44: OTG device IN endpoint x interrupt register (OTG_DIEPINTx) (x = 0..5, where x = Endpoint number)</a></li> <li>– <a href="#">Section 50.15.49: OTG device OUT endpoint x interrupt register (OTG_DOEPINTx) (x = 0..5, where x = Endpoint number)</a></li> <li>– <a href="#">Section 50.16.4: Host programming model</a></li> <li>– <a href="#">Figure 601: A-device SRP</a></li> <li>– <a href="#">Figure 602: B-device SRP</a></li> <li>– <a href="#">Table 344: Flexible TRACE pin assignment</a></li> </ul>

**Table 347. Document revision history (continued)**

Date	Revision	Changes
05-Mar-2018	3	<p>Added:</p> <ul style="list-style-type: none"> <li>– <a href="#">Section 22.3: DAC implementation</a></li> </ul> <p>Updated:</p> <ul style="list-style-type: none"> <li>– <a href="#">Section 1.2: List of abbreviations for registers</a></li> <li>– <a href="#">Section 11: Direct memory access controller (DMA)</a> and all subsections contents</li> <li>– <a href="#">Section 22.7: DAC registers</a>: naming conventions of all subsections</li> <li>– <a href="#">Section 22.7.1: DAC control register (DAC_CR)</a></li> <li>– <a href="#">Section 18.5.6: NOR/PSRAM controller registers</a></li> <li>– <a href="#">Section 18.6.7: NAND Flash controller registers</a></li> <li>– <a href="#">Table 109: FSMC register map</a></li> <li>– <a href="#">Section 28.7: LTDC registers</a>: naming conventions and reset value format of all subsections</li> <li>– <a href="#">Section 45.5.6: Interrupt and status register (LPUART_ISR)</a></li> <li>– <a href="#">Section 45.5.7: Interrupt flag clear register (LPUART_ICR)</a></li> <li>– <a href="#">Table 276: LPUART register map and reset values</a></li> <li>– <a href="#">Section 50.8.1: Host SOFs</a></li> <li>– <a href="#">Section 50.8.2: Peripheral SOFs</a></li> <li>– <a href="#">Section 50.11.3: FIFO RAM allocation</a></li> <li>– <a href="#">Section 50.15: OTG_FS registers</a>: naming conventions of all subsections</li> </ul> <p>Deleted:</p> <ul style="list-style-type: none"> <li>– From <a href="#">Section 12: DMA request multiplexer (DMAMUX)</a>: DMAMUX size identification register (DMAMUX_SIDR), DMAMUX IP identification register (DMAMUX_IPIDR), DMAMUX version register (DMAMUX_VERR), DMAMUX hardware configuration 1 register (DMAMUX_HWCFCGR1) and DMAMUX hardware configuration 2 register (DMAMUX_HWCFCGR2)</li> </ul>

**Table 347. Document revision history (continued)**

Date	Revision	Changes
03-May-2018	4	<p>Added:</p> <ul style="list-style-type: none"> <li>– <a href="#">Section 1.1: General information</a></li> <li>– <a href="#">Section 5.1.7: VDD12 domain</a> including <a href="#">Figure 9: Internal main regulator overview</a></li> <li>– <a href="#">Section Table 7.: Flash module - 1 Mbyte dual-bank organization (64 bits read width)</a></li> <li>– <a href="#">Section Table 8.: Flash module - 1 Mbyte single-bank organization (128 bits read width)</a></li> </ul> <p>Updated:</p> <ul style="list-style-type: none"> <li>– <a href="#">Section 3.3.1: Flash memory organization</a></li> <li>– <a href="#">Section 5.1.8: Dynamic voltage scaling management</a></li> <li>– DB1M bit description on <a href="#">Section 3.4.1: Option bytes description</a> and <a href="#">Section 3.7.8: Flash option register (FLASH_OPTR)</a></li> <li>– <a href="#">Figure 418</a> was moved to the created <a href="#">Section 41.3.1: WWDG block diagram</a></li> <li>– <a href="#">Section 8.4: GPIO registers</a>: naming conventions of all subsections</li> <li>– <a href="#">Figure 137: Dual-channel DAC block diagram</a></li> <li>– <a href="#">Section 22: Digital-to-analog converter (DAC)</a> naming conventions of all subsections</li> <li>– <a href="#">Section 22.4.1: DAC block diagram</a></li> <li>– <a href="#">Section 22.4.2: DAC channel enable</a></li> <li>– <a href="#">Section 22.4.8: Noise generation</a></li> <li>– <a href="#">Section 22.4.9: Triangle-wave generation</a></li> <li>– <a href="#">Section Table 132.: Sample and refresh timings</a></li> <li>– <a href="#">Section : Example of the sample and refresh time calculation with output buffer on on page 679</a></li> <li>– <a href="#">Section 47.5: SAI registers</a> was fully restructured</li> <li>– <a href="#">Figure 48: Secure digital input/output MultiMediaCard interface (SDMMC)</a> naming conventions of all subsections</li> <li>– <a href="#">Figure 536: SDMMC block diagram</a></li> <li>– <a href="#">Table 292: SDMMC internal input/output signals</a></li> <li>– <a href="#">Table 293: SDMMC pins</a></li> <li>– <a href="#">Figure 48.4.3: General description</a></li> <li>– <a href="#">Table 294: SDMMC Command and data phase selection</a></li> <li>– <a href="#">Table 294: SDMMC Command and data phase selection</a></li> <li>– <a href="#">Figure 538: Control unit</a></li> <li>– <a href="#">Table 311: CMD12 use cases</a></li> </ul>

# Index

## A

ADC_AWD2CR	660
ADC_AWD3CR	660
ADC_CALFACT	661
ADC_CCR	663
ADC_CFGR	643
ADC_CFGR2	647
ADC_CR	640
ADC_CSR	662
ADC_DIFSEL	661
ADC_DR	656
ADC_IER	638
ADC_ISR	636
ADC_JDRy	659
ADC_JSQR	657
ADC_OFRy	658
ADC_SMPR1	649
ADC_SMPR2	650
ADC_SQR1	653
ADC_SQR2	654
ADC_SQR3	655
ADC_SQR4	656
ADC_TR1	650
ADC_TR2	651
ADC_TR3	652
AES_CR	1046
AES_DINR	1050
AES_DOUTR	1051
AES_IVR0	1053
AES_IVR1	1053
AES_IVR2	1053
AES_IVR3	1054
AES_KEYR1	1052
AES_KEYR2	1052
AES_KEYR3	1052
AES_KEYR4	1054
AES_KEYR5	1054
AES_KEYR6	1055
AES_KEYR7	1055
AES_KEYRx	1051
AES_SR	1048
AES_SUSPxR	1055

## C

CAN_BTR	1848
CAN_ESR	1847
CAN_FA1R	1858

CAN_FFA1R	1857
CAN_FiRx	1859
CAN_FM1R	1856
CAN_FMR	1856
CAN_FS1R	1857
CAN_IER	1846
CAN_MCR	1839
CAN_MSR	1841
CAN_RDHxR	1855
CAN_RDLxR	1855
CAN_RDTxR	1854
CAN_RF0R	1844
CAN_RF1R	1845
CAN_RIxR	1853
CAN_TDHxR	1852
CAN_TDLxR	1852
CAN_TDTxR	1851
CAN_TIxR	1850
CAN_TSR	1842
COMP1_CSR	738
COMP2_CSR	740
CRC_CR	458
CRC_DR	457
CRC_IDR	457
CRC_INIT	458
CRC_POL	459
CRS_CFGR	302
CRS_CR	301
CRS_ICR	305
CRS_ISR	303

## D

DAC_CCR	697
DAC_CR	687
DAC_DHR12L1	692
DAC_DHR12L2	693
DAC_DHR12LD	694
DAC_DHR12R1	691
DAC_DHR12R2	692
DAC_DHR12RD	694
DAC_DHR8R1	692
DAC_DHR8R2	693
DAC_DHR8RD	694
DAC_DOR1	695
DAC_DOR2	695
DAC_MCR	697
DAC_SHHR	699
DAC_SHRR	700

DAC_SHSR1	699	DMAMUX_RGxCR	381
DAC_SHSR2	699	DSIHOST_CCR	913
DAC_SR	696	DSIHOST_CLCR	931
DAC_SWTRGR	690	DSIHOST_CLTCR	932
DBGMCU_APB1FZR1	2040	DSIHOST_CMCR	924
DBGMCU_APB1FZR2	2042	DSIHOST_CR	912
DBGMCU_APB2FZR	2042	DSIHOST_DLTCCR	932
DBGMCU_CR	2039	DSIHOST_FIR0	942
DBGMCU_IDCODE	2026	DSIHOST_FIR1	944
DCMI_CR	715	DSIHOST_GHCR	926
DCMI_CWSIZE	726	DSIHOST_GPDR	926
DCMI_CWSTRT	726	DSIHOST_GPSR	927
DCMI_DR	727	DSIHOST_GVCIDR	916
DCMI_ESCR	724	DSIHOST_IER0	938
DCMI_ESUR	725	DSIHOST_IER1	941
DCMI_ICR	723	DSIHOST_ISR0	936
DCMI_IER	721	DSIHOST_ISR1	937
DCMI_MIS	722	DSIHOST_LCCR	923
DCMI_RIS	720	DSIHOST_LCOLCR	914
DCMI_SR	719	DSIHOST_LCVCIDR	945
DFSDM_CHyAWSCDR	789	DSIHOST_LPCR	914
DFSDM_CHyCFGR1	786	DSIHOST_LPMCCR	946
DFSDM_CHyCFGR2	788	DSIHOST_LPMCR	915
DFSDM_CHyDATINR	790	DSIHOST_LVCIDR	913
DFSDM_CHyDLyR	791	DSIHOST_MCR	917
DFSDM_CHyWDATR	790	DSIHOST_PCONFR	933
DFSDM_FLTxAWCFR	804	DSIHOST_PCR	915
DFSDM_FLTxAWHTR	802	DSIHOST_PCTLR	933
DFSDM_FLTxAWLTR	802	DSIHOST_PSR	935
DFSDM_FLTxAWSR	803	DSIHOST_PTTCCR	935
DFSDM_FLTxCNVTIMR	805	DSIHOST_PUCR	934
DFSDM_FLTxCR1	792	DSIHOST_TCCR1	928
DFSDM_FLTxCR2	795	DSIHOST_TCCR2	928
DFSDM_FLTxEXMAX	804	DSIHOST_TCCR3	929
DFSDM_FLTxEXMIN	805	DSIHOST_TCCR4	929
DFSDM_FLTxFCR	799	DSIHOST_TCCR5	930
DFSDM_FLTxICR	798	DSIHOST_TCCR6	931
DFSDM_FLTxISR	796	DSIHOST_VCCCR	949
DFSDM_FLTxJCHGR	799	DSIHOST_VCCR	919
DFSDM_FLTxJDATAR	800	DSIHOST_VHBPCCR	950
DFSDM_FLTxRDATAR	801	DSIHOST_VHBPCR	921
DMA_CCRx	362	DSIHOST_VHSACCR	950
DMA_CMARx	366	DSIHOST_VHSACR	920
DMA_CNDTRx	365	DSIHOST_VLCCR	951
DMA_CPARx	365	DSIHOST_VLCR	921
DMA_IFCR	361	DSIHOST_VMCCR	947
DMA_ISR	358	DSIHOST_VMCR	917
DMAMUX_CFR	380	DSIHOST_VNPCCR	949
DMAMUX_CSR	380	DSIHOST_VPCCR	948
DMAMUX_CxCR	379	DSIHOST_VPCR	919
DMAMUX_RGCFR	382	DSIHOST_VR	912
DMAMUX_RGSR	382	DSIHOST_VSCR	945

DSIHOST_VVACCR	953	FSMC_BCRx	496
DSIHOST_VVACR	923	FSMC_BTRx	499
DSIHOST_VVBPCR	952	FSMC_BWTR1..4	502
DSIHOST_VVBPCR	922	FSMC_ECCR	515
DSIHOST_VVFPCCR	952	FSMC_PATT	513
DSIHOST_VVFPCCR	922	FSMC_PCR	510
DSIHOST_VVSACCR	951	FSMC_PMEM	512
DSIHOST_VVSACR	921	FSMC_SR	511
DSIWRAP_CFGR	954	FW_CR	160
DSIWRAP_CR	955	FW_CSL	157
DSIWRAP_IER	956	FW_CSSA	157
DSIWRAP_IFCR	958	FW_NVDSL	158
DSIWRAP_ISR	957	FW_NVSSA	158
DSIWRAP_PCR1	959	FW_VDSL	159
DSIWRAP_PCR2	961	FW_VSSA	159
DSIWRAP_PCR3	963		
DSIWRAP_PCR4	964		
DSIWRAP_PCR5	965		
DSIWRAP_RPCR	965		
<b>E</b>			
EXTI_EMR1	446	GFXMMU_CR	429
EXTI_EMR2	450	GPIOx_AFRH	321
EXTI_FTSR1	447	GPIOx_AFRL	320
EXTI_FTSR2	451	GPIOx_BRR	322
EXTI_IMR1	446	GPIOx_BSRR	319
EXTI_IMR2	449	GPIOx_IDR	318
EXTI_PR1	449	GPIOx_LCKR	319
EXTI_PR2	452	GPIOx_MODER	316
EXTI_RTSR1	446	GPIOx_ODR	318
EXTI_RTSR2	450	GPIOx_OSPEEDR	317
EXTI_SWIER1	448	GPIOx_OTYPER	317
EXTI_SWIER2	451	GPIOx_PUPDR	317
<b>F</b>			
FLASH_ACR	131		
FLASH_CR	136		
FLASH_ECCR	138		
FLASH_KEYR	133		
FLASH_OPTKEYR	133		
FLASH_OPTR	140		
FLASH_PCROP1ER	142		
FLASH_PCROP1SR	142		
FLASH_PCROP2ER	145		
FLASH_PCROP2SR	144		
FLASH_PDKEYR	132		
FLASH_SR	134		
FLASH_WRP1AR	143		
FLASH_WRP1BR	144		
FLASH_WRP2AR	145		
FLASH_WRP2BR	146		
<b>G</b>			
<b>H</b>			
		HASH_CR	1071
		HASH_CSR0	1081
		HASH_CSRx	1081
		HASH_DIN	1074
		HASH_HR0	1076
		HASH_HR1	1076
		HASH_HR2	1077
		HASH_HR3	1077
		HASH_HR4	1077
		HASH_HR5	1078
		HASH_HR6	1078
		HASH_HR7	1078
		HASH_IMR	1079
		HASH_SR	1080
		HASH_STR	1075
<b>I</b>			
		I2C_CR1	1498
		I2C_CR2	1501

I2C_ICR	1510	LTDC_LxCFBLR	845
I2C_ISR	1508	LTDC_LxCKCR	841
I2C_OAR1	1504	LTDC_LxCLUTWR	847
I2C_OAR2	1505	LTDC_LxCR	838
I2C_PECR	1511	LTDC_LxDCCR	843
I2C_RXDR	1512	LTDC_LxPFCR	841
I2C_TIMEOUTR	1507	LTDC_LxWHPCR	839
I2C_TIMINGR	1506	LTDC_LxWVPCR	840
I2C_TXDR	1512	LTDC_SRCR	833
I2Cx_CR2	157-158, 160, 1501	LTDC_SSCR	829
IWDG_KR	1388	LTDC_TWCR	831
IWDG_PR	1389		
IWDG_RLR	1390		
IWDG_SR	1391		
IWDG_WINR	1392		
<b>L</b>			
LPTIM_ARR	1380		
LPTIM_CFGR	1375		
LPTIM_CMP	1379		
LPTIM_CNT	1380		
LPTIM_CR	1378		
LPTIM_ICR	1373		
LPTIM_IER	1374		
LPTIM_ISR	1372		
LPTIM1_OR	1381		
LPTIM2_OR	1381		
LPUART_BRR	1630		
LPUART_CR1	1622		
LPUART_CR2	1625		
LPUART_CR3	1627		
LPUART_ICR	1635		
LPUART_ISR	1631		
LPUART_PRESC	1637		
LPUART_RDR	1636		
LPUART_RQR	1631		
LPUART_TDR	1637		
LTDC_AWCR	830		
LTDC_BCCR	833		
LTDC_BPCR	829		
LTDC_CDSR	837		
LTDC_CPSR	836		
LTDC_GCR	831		
LTDC_ICR	835		
LTDC_IER	834		
LTDC_ISR	835		
LTDC_LIPCR	836		
LTDC_LxBFCR	844		
LTDC_LxCACR	842		
LTDC_LxCFBAR	845		
LTDC_LxCFBLNR	846		
		<b>O</b>	
		OCTOSPI_PIR	554
		OCTOSPI_PSMAR	554
		OCTOSPI_PSMKR	553
		OCTOSPI_ABR	558
		OCTOSPI_AR	552
		OCTOSPI_CCR	554
		OCTOSPI_CR	545
		OCTOSPI_DCR1	547
		OCTOSPI_DCR2	549
		OCTOSPI_DCR3	549
		OCTOSPI_DLR	552
		OCTOSPI_DR	553
		OCTOSPI_FCR	551
		OCTOSPI_IR	558
		OCTOSPI_LPTR	559
		OCTOSPI_TCR	557
		OCTOSPI_WCCR	559
		OCTOSPIOM_PnCR)	570
		OPAMP1_CSR	752
		OPAMP1_LPOTR	753-755
		OPAMP1_OTR	753
		OTG_CID	1917
		OTG_DAIN	1942
		OTG_DAINMSK	1943
		OTG_DCFG	1935
		OTG_DCTL	1936
		OTG_DIEPCTL0	1945
		OTG_DIEPCTLx	1946
		OTG_DIEPEMPMSK	1944
		OTG_DIEPINTx	1949
		OTG_DIEPMSK	1940
		OTG_DIEPTSIZ0	1950
		OTG_DIEPTSIZx	1952
		OTG_DIEPTXF0	1913
		OTG_DIEPTXFx	1924
		OTG_DOEPCTL0	1953
		OTG_DOEPCTLx	1957
		OTG_DOEPINTx	1954

OTG_DOEPMSK	1941
OTG_DOEPTSIZE0	1956
OTG_DOEPTSIZEx	1959
OTG_DSTS	1939
OTG_DTXFSTSx	1951
OTG_DVBUSDIS	1943
OTG_DVBUSPULSE	1944
OTG_GADPCTL	1921
OTG_GAHBCFG	1898
OTG_GCCFG	1915
OTG_GINTMSK	1908
OTG_GINTSTS	1904
OTG_GLPMCFG	1917
OTG_GOTGCTL	1894
OTG_GOTGINT	1897
OTG_GPWRDN	1921
OTG_GRSTCTL	1901
OTG_GRXFSIZ	1913
OTG_GRXSTSP	1911
OTG_GRXSTSR	1911
OTG_GUSBCFG	1899
OTG_HAINT	1928
OTG_HAINTMSK	1928
OTG_HCCHARx	1931
OTG_HCFG	1924
OTG_HCINTMSKx	1933
OTG_HCINTx	1932
OTG_HCTSIZx	1934
OTG_HFIR	1925
OTG_HFNUM	1926
OTG_HNPTXFSIZ	1913
OTG_HNPTXSTS	1914
OTG_HPRT	1929
OTG_HPTXFSIZ	1923
OTG_HPTXSTS	1927
OTG_PCGCCTL	1960

**P**

purpose	1260
PWR_CR1	196
PWR_CR2	197
PWR_CR3	198
PWR_CR4	199
PWR_PDCRA	204
PWR_PDCRB	205
PWR_PDCRC	206
PWR_PDCRD	207
PWR_PDCRE	208
PWR_PDCRF	209, 212
PWR_PDCRG	210
PWR_PDCRH	211

PWR_PUCRA	204
PWR_PUCRB	205
PWR_PUCRC	206
PWR_PUCRD	207
PWR_PUCRE	208
PWR_PUCRF	208, 211
PWR_PUCRG	209
PWR_PUCRH	210
PWR_SCR	203
PWR_SR1	200
PWR_SR2	201

**R**

RCC_AHB1ENR	263
RCC_AHB1RSTR	254
RCC_AHB1SMENR	272
RCC_AHB2ENR	264
RCC_AHB2RSTR	255
RCC_AHB2SMENR	274
RCC_AHB3ENR	266
RCC_AHB3RSTR	257
RCC_AHB3SMENR	276
RCC_APB1ENR1	267
RCC_APB1ENR2	269
RCC_APB1RSTR1	258
RCC_APB1RSTR2	260
RCC_APB1SMENR1	277
RCC_APB1SMENR2	279
RCC_APB2ENR	271
RCC_APB2RSTR	261
RCC_APB2SMENR	281
RCC_BDCR	285
RCC_CCIPR	282
RCC_CFGR	238
RCC_CICR	252
RCC_CIER	249
RCC_CIFR	251
RCC_CR	235
RCC_CRRCR	288
RCC_CSR	286
RCC_ICSCR	238
RCC_PLLCFGR	241
RCC_PLLSAI1CFGR	243
RCC_PLLSAI2CFGR	247
RNG_CR	1001
RNG_DR	1003
RNG_SR	1002
RTC_ALRMAR	1430
RTC_ALRMBR	1431
RTC_ALRMBSSR	1442
RTC_BKPxR	1443



RTC_CALR	1437	SDMMC_RESPCMDR	1800
RTC_CR	1422	SDMMC_RESPxR	1801
RTC_DR	1421	SDMMC_STAR	1805
RTC_ISR	1425	SPIx_CR1	1666
RTC_OR	1443	SPIx_CR2	1668
RTC_PRER	1428	SPIx_CRCPR	1671
RTC_SHIFTR	1433	SPIx_DR	1671
RTC_SSR	1432	SPIx_RXCRCR	1672
RTC_TR	1420	SPIx_SR	1670
RTC_TSDR	1435	SPIx_TXCRCR	1672
RTC_TSSSR	1436	SYSCFG_CFGR1	326
RTC_TSTR	1434	SYSCFG_CFGR2	335
RTC_WPR	1432	SYSCFG_EXTICR1	328
RTC_WUTR	1429	SYSCFG_EXTICR2	330
		SYSCFG_EXTICR3	331
		SYSCFG_EXTICR4	333
		SYSCFG_MEMRMP	325
		SYSCFG_SCSR	334
		SYSCFG_SKR	336
		SYSCFG_SWPR	336-337
<b>S</b>			
SAI_ACLRFR	1733		
SAI_ACR1	1712		
SAI_ACR2	1718		
SAI_ADR	1735		
SAI_AFRCR	1722		
SAI_AIM	1726		
SAI_ASLOTR	1724		
SAI_ASR	1729		
SAI_BCLRFR	1734		
SAI_BCR1	1715		
SAI_BCR2	1720		
SAI_BDR	1736		
SAI_BFRCR	1723		
SAI_BIM	1728		
SAI_BSLOTR	1725		
SAI_BSR	1731		
SAI_GCR	1712		
SAI_PDMCR	1736		
SAI_PDMDLY	1737		
SDMMC_ACKTIMER	1812		
SDMMC_ARGR	1798		
SDMMC_CLKCR	1796		
SDMMC_CMDR	1798		
SDMMC_DCNTR	1804		
SDMMC_DCTRL	1803		
SDMMC_DLENR	1802		
SDMMC_DTIMER	1801		
SDMMC_FIFOR	1813		
SDMMC_ICR	1807		
SDMMC_IDMABASE0R	1815		
SDMMC_IDMABASE1R	1815		
SDMMC_IDMABSIZER	1814		
SDMMC_IDMACTRLR	1813		
SDMMC_MASKR	1810		
SDMMC_POWER	1795		
		<b>T</b>	
		TIM1_OR1	1173
		TIM1_OR2	1176
		TIM1_OR3	1178
		TIM15_ARR	1314
		TIM15_BDTR	1316
		TIM15_CCER	1311
		TIM15_CCMR1	1307
		TIM15_CCR1	1315
		TIM15_CCR2	1316
		TIM15_CNT	1314
		TIM15_CR1	1299
		TIM15_CR2	1300
		TIM15_DCR	1319
		TIM15_DIER	1303
		TIM15_DMAR	1319
		TIM15_EGR	1306
		TIM15_OR1	1320
		TIM15_OR2	1320
		TIM15_PSC	1314
		TIM15_RCR	1315
		TIM15_SMCR	1302
		TIM15_SR	1304
		TIM16_OR1	1340
		TIM16_OR2	1341
		TIM17_OR1	1342
		TIM17_OR2	1343
		TIM2_OR1	1255
		TIM2_OR2	1256
		TIM3_OR1	1255

TIM3_OR2	1256
TIM8_OR1	1173
TIM8_OR2	1179
TIM8_OR3	1181
TIMx_ARR	1164, 1251, 1335, 1358
TIMx_BDTR	1167, 1337
TIMx_CCER	1160, 1249, 1332
TIMx_CCMR1	1154, 1243, 1330
TIMx_CCMR2	1158, 1247
TIMx_CCMR3	1174
TIMx_CCR1	1165, 1251, 1336
TIMx_CCR2	1166, 1252
TIMx_CCR3	1166, 1252
TIMx_CCR4	1167, 1253
TIMx_CCR5	1175
TIMx_CCR6	1176
TIMx_CNT	1164, 1250, 1334, 1357
TIMx_CR1	1143, 1234, 1325, 1354
TIMx_CR2	1144, 1235, 1326, 1356
TIMx_DCR	1171, 1254, 1339
TIMx_DIER	1149, 1240, 1327, 1356
TIMx_DMAR	1172, 1254, 1340
TIMx_EGR	1153, 1242, 1329, 1357
TIMx_PSC	1164, 1251, 1335, 1358
TIMx_RCR	1165, 1336
TIMx_SMCR	1147, 1237
TIMx_SR	1151, 1241, 1328, 1357
TSC_CR	981
TSC_ICR	984
TSC_IER	983
TSC_IOASCR	986
TSC_IOCCR	987
TSC_IQGCSR	987
TSC_IQGXCR	988
TSC_IQHCR	985
TSC_IOSCR	986
TSC_ISR	985

**U**

USART_BRR	1579
USART_CR1	1566
USART_CR2	1571
USART_CR3	1574
USART_GTPR	1579
USART_ICR	1588
USART_ISR	1582
USART_PRESC	1591
USART_RDR	1590
USART_RQR	1581
USART_RTOR	1580
USART_TDR	1590

**V**

VREFBUF_CCR	731
VREFBUF_CSR	730

**W**

WWDG_CFR	1398
WWDG_CR	1398
WWDG_SR	1399

**IMPORTANT NOTICE – PLEASE READ CAREFULLY**

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2018 STMicroelectronics – All rights reserved

