RM0090
Reference manual

STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced Arm®-based 32-bit MCUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32F405xx/07xx, STM32F415xx/17xx, STM32F42xxx and STM32F43xxx microcontroller memory and peripherals.

The STM32F405xx/07xx, STM32F415xx/17xx, STM32F42xxx and STM32F43xxx constitute a family of microcontrollers with different memory sizes, packages and peripherals.

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

For information on the Arm® Cortex®-M4 with FPU core, please refer to the *Cortex®-M4 with FPU Technical Reference Manual*.

Related documents

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

- STM32F40x and STM32F41x datasheets
- STM32F42x and STM32F43x datasheets
- For information on the Arm® *Cortex®*-M4 with FPU*, refe*r to the *STM32F3xx/F4xxx Cortex®*-M4 with FPU *programming manual (PM0214).*

Contents

1 Documentation conventions 57

1.1 List of abbreviations for registers 57

1.2 Glossary 58

1.3 Peripheral availability 58

2 Memory and bus architecture 59

2.1 System architecture 59

2.1.1 I-bus 62

2.1.2 D-bus 62

2.1.3 S-bus 62

2.1.4 DMA memory bus 63

2.1.5 DMA peripheral bus 63

2.1.6 Ethernet DMA bus 63

2.1.7 USB OTG HS DMA bus 63

2.1.8 LCD-TFT controller DMA bus 63

2.1.9 DMA2D bus 63

2.1.10 BusMatrix 63

2.1.11 AHB/APB bridges (APB) 64

2.2 Memory organization 64

2.3 Memory map 64

2.3.1 Embedded SRAM 68

2.3.2 Flash memory overview 68

2.3.3 Bit banding 68

2.4 Boot configuration 69

3 Embedded Flash memory interface 73

3.1 Introduction 73

3.2 Main features 73

3.3 Embedded Flash memory in STM32F405xx/07xx and STM32F415xx/17xx 74

3.4 Embedded Flash memory in STM32F42xxx and STM32F43xxx 76

3.5 Read interface 80

3.5.1 Relation between CPU clock frequency and Flash memory read time . 80

3.5.2 Adaptive real-time memory accelerator (ART Accelerator™) 82
3.6 Erase and program operations 84
3.6.1 Unlocking the Flash control register 84
3.6.2 Program/erase parallelism 85
3.6.3 Erase 85
3.6.4 Programming 86
3.6.5 Read-while-write (RWW) 87
3.6.6 Interrupts 88
3.7 Option bytes 88
3.7.1 Description of user option bytes 88
3.7.2 Programming user option bytes 92
3.7.3 Read protection (RDP) 93
3.7.4 Write protections 94
3.7.5 Proprietary code readout protection (PCROP) 95
3.8 One-time programmable bytes 97
3.9 Flash interface registers 98
3.9.1 Flash access control register (FLASH_ACR) for STM32F405xx/07xx and STM32F415xx/17xx 98
3.9.2 Flash access control register (FLASH_ACR) for STM32F42xxx and STM32F43xxx 99
3.9.3 Flash key register (FLASH_KEYR) 100
3.9.4 Flash option key register (FLASH_OPTKEYR) 100
3.9.5 Flash status register (FLASH_SR) for STM32F405xx/07xx and STM32F415xx/17xx 101
3.9.6 Flash status register (FLASH_SR) for STM32F42xxx and STM32F43xxx 102
3.9.7 Flash control register (FLASH_CR) for STM32F405xx/07xx and STM32F415xx/17xx 103
3.9.8 Flash control register (FLASH_CR) for STM32F42xxx and STM32F43xxx 105
3.9.9 Flash option control register (FLASH_OPTCR) for STM32F405xx/07xx and STM32F415xx/17xx 106
3.9.10 Flash option control register (FLASH_OPTCR) for STM32F42xxx and STM32F43xxx 108
3.9.11 Flash option control register (FLASH_OPTCR1) for STM32F42xxx and STM32F43xxx 110
3.9.12 Flash interface register map 111

4 CRC calculation unit 113

4.1 CRC introduction 113
4.2 CRC main features 113
4.3 CRC functional description 114
4.4 CRC registers 114
4.4.1 Data register (CRC_DR) 114
4.4.2 Independent data register (CRC_IDR) 114
4.4.3 Control register (CRC_CR) 115
4.4.4 CRC register map 115

5 Power controller (PWR) 116
5.1 Power supplies 116
5.1.1 Independent A/D converter supply and reference voltage 117
5.1.2 Battery backup domain 118
5.1.3 Voltage regulator for STM32F405xx/07xx and STM32F415xx/17xx . . 120
5.1.4 Voltage regulator for STM32F42xxx and STM32F43xxx 121
5.2 Power supply supervisor 124
5.2.1 Power-on reset (POR)/power-down reset (PDR) 124
5.2.2 Brownout reset (BOR) 125
5.2.3 Programmable voltage detector (PVD) 125
5.3 Low-power modes 126
5.3.1 Slowing down system clocks 128
5.3.2 Peripheral clock gating 128
5.3.3 Sleep mode 129
5.3.4 Stop mode (STM32F405xx/07xx and STM32F415xx/17xx) 130
5.3.5 Stop mode (STM32F42xxx and STM32F43xxx) 133
5.3.6 Standby mode 136
5.3.7 Programming the RTC alternate functions to wake up the device from the Stop and Standby modes 138
5.4 Power control registers (STM32F405xx/07xx and STM32F415xx/17xx) 141
5.4.1 PWR power control register (PWR_CR) for STM32F405xx/07xx and STM32F415xx/17xx 141
5.4.2 PWR power control/status register (PWR_CSR) for STM32F405xx/07xx and STM32F415xx/17xx 142
5.5 Power control registers (STM32F42xxx and STM32F43xxx) 144
5.5.1 PWR power control register (PWR_CR) for STM32F42xxx and STM32F43xxx 144
5.5.2 PWR power control/status register (PWR_CSR) for STM32F42xxx and STM32F43xxx 147

5.6 PWR register map 149

6 Reset and clock control for STM32F42xxx and STM32F43xxx (RCC) 150

6.1 Reset 150
6.1.1 System reset 150
6.1.2 Power reset 150
6.1.3 Backup domain reset 151
6.2 Clocks 151
6.2.1 HSE clock 154
6.2.2 HSI clock 155
6.2.3 PLL configuration 155
6.2.4 LSE clock 156
6.2.5 LSI clock 156
6.2.6 System clock (SYSCLK) selection 156
6.2.7 Clock security system (CSS) 157
6.2.8 RTC/AWU clock 157
6.2.9 Watchdog clock 158
6.2.10 Clock-out capability 158
6.2.11 Internal/external clock measurement using TIM5/TIM11 158
6.3 RCC registers 161
6.3.1 RCC clock control register (RCC_CR) 161
6.3.2 RCC PLL configuration register (RCC_PLLCFGR) 163
6.3.3 RCC clock configuration register (RCC_CFGR) 165
6.3.4 RCC clock interrupt register (RCC_CIR) 167
6.3.5 RCC AHB1 peripheral reset register (RCC_AHB1RSTR) 170
6.3.6 RCC AHB2 peripheral reset register (RCC_AHB2RSTR) 173
6.3.7 RCC AHB3 peripheral reset register (RCC_AHB3RSTR) 174
6.3.8 RCC APB1 peripheral reset register (RCC_APB1RSTR) 174
6.3.9 RCC APB2 peripheral reset register (RCC_APB2RSTR) 178
6.3.10 RCC AHB1 peripheral clock register (RCC_AHB1ENR) 180
6.3.11 RCC AHB2 peripheral clock enable register (RCC_AHB2ENR) 182
6.3.12 RCC AHB3 peripheral clock enable register (RCC_AHB3ENR) 183
6.3.13 RCC APB1 peripheral clock enable register (RCC_APB1ENR) 183
6.3.14 RCC APB2 peripheral clock enable register (RCC_APB2ENR) 187
6.3.15 RCC AHB1 peripheral clock enable in low power mode register (RCC_AHB1LPENR) 189

6.3.16 RCC AHB2 peripheral clock enable in low power mode register (RCC_AHB2LPENR) . . . 192
6.3.17 RCC AHB3 peripheral clock enable in low power mode register (RCC_AHB3LPENR) . . . 193
6.3.18 RCC APB1 peripheral clock enable in low power mode register (RCC_APB1LPENR) . . . 193
6.3.19 RCC APB2 peripheral clock enabled in low power mode register (RCC_APB2LPENR) . . . 197
6.3.20 RCC Backup domain control register (RCC_BDCR) . . . 199
6.3.21 RCC clock control & status register (RCC_CSR) . . . 200
6.3.22 RCC spread spectrum clock generation register (RCC_SSCGR) . . . 202
6.3.23 RCC PLLI2S configuration register (RCC_PLLI2SCFGR) . . . 203
6.3.24 RCC PLL configuration register (RCC_PLLSAICFGR) . . . 206
6.3.25 RCC Dedicated Clock Configuration Register (RCC_DCKCFGR) . . . 207
6.3.26 RCC register map . . . 210

7 Reset and clock control for STM32F405xx/07xx and STM32F415xx/17xx(RCC) . . . 213

7.1 Reset . . . 213
7.1.1 System reset . . . 213
7.1.2 Power reset . . . 214
7.1.3 Backup domain reset . . . 214
7.2 Clocks . . . 215
7.2.1 HSE clock . . . 217
7.2.2 HSI clock . . . 218
7.2.3 PLL configuration . . . 219
7.2.4 LSE clock . . . 219
7.2.5 LSI clock . . . 220
7.2.6 System clock (SYSCLK) selection . . . 220
7.2.7 Clock security system (CSS) . . . 220
7.2.8 RTC/AWU clock . . . 221
7.2.9 Watchdog clock . . . 221
7.2.10 Clock-out capability . . . 222
7.2.11 Internal/external clock measurement using TIM5/TIM11 . . . 222
7.3 RCC registers . . . 224
7.3.1 RCC clock control register (RCC_CR) . . . 224
7.3.2 RCC PLL configuration register (RCC_PLLCFGR) . . . 226
7.3.3 RCC clock configuration register (RCC_CFGR) . . . 228

7.3.4 RCC clock interrupt register (RCC_CIR) 230
7.3.5 RCC AHB1 peripheral reset register (RCC_AHB1RSTR) 233
7.3.6 RCC AHB2 peripheral reset register (RCC_AHB2RSTR) 236
7.3.7 RCC AHB3 peripheral reset register (RCC_AHB3RSTR) 237
7.3.8 RCC APB1 peripheral reset register (RCC_APB1RSTR) 237
7.3.9 RCC APB2 peripheral reset register (RCC_APB2RSTR) 240
7.3.10 RCC AHB1 peripheral clock enable register (RCC_AHB1ENR) 242
7.3.11 RCC AHB2 peripheral clock enable register (RCC_AHB2ENR) 244
7.3.12 RCC AHB3 peripheral clock enable register (RCC_AHB3ENR) 245
7.3.13 RCC APB1 peripheral clock enable register (RCC_APB1ENR) 245
7.3.14 RCC APB2 peripheral clock enable register (RCC_APB2ENR) 248
7.3.15 RCC AHB1 peripheral clock enable in low power mode register (RCC_AHB1LPENR) 250
7.3.16 RCC AHB2 peripheral clock enable in low power mode register (RCC_AHB2LPENR) 252
7.3.17 RCC AHB3 peripheral clock enable in low power mode register (RCC_AHB3LPENR) 253
7.3.18 RCC APB1 peripheral clock enable in low power mode register (RCC_APB1LPENR) 254
7.3.19 RCC APB2 peripheral clock enabled in low power mode register (RCC_APB2LPENR) 257
7.3.20 RCC Backup domain control register (RCC_BDCR) 259
7.3.21 RCC clock control & status register (RCC_CSR) 260
7.3.22 RCC spread spectrum clock generation register (RCC_SSCGR) 262
7.3.23 RCC PLLI2S configuration register (RCC_PLLI2SCFGR) 263
7.3.24 RCC register map 265

8 General-purpose I/Os (GPIO) 267

8.1 GPIO introduction 267
8.2 GPIO main features 267
8.3 GPIO functional description 267
8.3.1 General-purpose I/O (GPIO) 269
8.3.2 I/O pin multiplexer and mapping 270
8.3.3 I/O port control registers 274
8.3.4 I/O port data registers 274
8.3.5 I/O data bitwise handling 274
8.3.6 GPIO locking mechanism 274
8.3.7 I/O alternate function input/output 275

8.3.8 External interrupt/wakeup lines . . . 275
8.3.9 Input configuration . . . 275
8.3.10 Output configuration . . . 276
8.3.11 Alternate function configuration . . . 277
8.3.12 Analog configuration . . . 278
8.3.13 Using the OSC32_IN/OSC32_OUT pins as GPIO PC14/PC15 port pins . . . 278
8.3.14 Using the OSC_IN/OSC_OUT pins as GPIO PH0/PH1 port pins . . . 278
8.3.15 Selection of RTC_AF1 and RTC_AF2 alternate functions . . . 279
8.4 GPIO registers . . . 281
8.4.1 GPIO port mode register (GPIOx_MODER) (x = A..I/J/K) . . . 281
8.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A..I/J/K) . . . 281
8.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A..I/J/K) . . . 282
8.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A..I/J/K) . . . 282
8.4.5 GPIO port input data register (GPIOx_IDR) (x = A..I/J/K) . . . 283
8.4.6 GPIO port output data register (GPIOx_ODR) (x = A..I/J/K) . . . 283
8.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..I/J/K) . . . 284
8.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A..I/J/K) . . . 284
8.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A..I/J/K) . . . 285
8.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A..I/J) . . . 286
8.4.11 GPIO register map . . . 287

9 System configuration controller (SYSCFG) . . . 289
9.1 I/O compensation cell . . . 289
9.2 SYSCFG registers for STM32F405xx/07xx and STM32F415xx/17xx . . . 289
9.2.1 SYSCFG memory remap register (SYSCFG_MEMRMP) . . . 289
9.2.2 SYSCFG peripheral mode configuration register (SYSCFG_PMC) . . . 290
9.2.3 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1) . . . 291
9.2.4 SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2) . . . 291
9.2.5 SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3) . . . 292
9.2.6 SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4) . . . 293

9.2.7 Compensation cell control register (SYSCFG_CMPCR) 293
9.2.8 SYSCFG register maps for STM32F405xx/07xx and STM32F415xx/17xx 294
9.3 SYSCFG registers for STM32F42xxx and STM32F43xxx 294
9.3.1 SYSCFG memory remap register (SYSCFG_MEMRMP) 294
9.3.2 SYSCFG peripheral mode configuration register (SYSCFG_PMC) . . 296
9.3.3 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1) 297
9.3.4 SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2) 298
9.3.5 SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3) 298
9.3.6 SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4) 299
9.3.7 Compensation cell control register (SYSCFG_CMPCR) 300
9.3.8 SYSCFG register maps for STM32F42xxx and STM32F43xxx 301

10 DMA controller (DMA) 302
10.1 DMA introduction 302
10.2 DMA main features 302
10.3 DMA functional description 304
10.3.1 General description 304
10.3.2 DMA transactions 306
10.3.3 Channel selection 307
10.3.4 Arbiter 308
10.3.5 DMA streams 309
10.3.6 Source, destination and transfer modes 309
10.3.7 Pointer incrementation 312
10.3.8 Circular mode 313
10.3.9 Double buffer mode 313
10.3.10 Programmable data width, packing/unpacking, endianess 314
10.3.11 Single and burst transfers 316
10.3.12 FIFO 317
10.3.13 DMA transfer completion 319
10.3.14 DMA transfer suspension 320
10.3.15 Flow controller 321
10.3.16 Summary of the possible DMA configurations 322
10.3.17 Stream configuration procedure 322

10.3.18 Error management . . . 323
10.4 DMA interrupts . . . 324
10.5 DMA registers . . . 325
10.5.1 DMA low interrupt status register (DMA_LISR) . . . 325
10.5.2 DMA high interrupt status register (DMA_HISR) . . . 326
10.5.3 DMA low interrupt flag clear register (DMA_LIFCR) . . . 327
10.5.4 DMA high interrupt flag clear register (DMA_HIFCR) . . . 327
10.5.5 DMA stream x configuration register (DMA_SxCR) (x = 0..7) . . . 328
10.5.6 DMA stream x number of data register (DMA_SxNDTR) (x = 0..7) . . . 331
10.5.7 DMA stream x peripheral address register (DMA_SxPAR) (x = 0..7) . 332
10.5.8 DMA stream x memory 0 address register (DMA_SxM0AR) (x = 0..7) 332
10.5.9 DMA stream x memory 1 address register (DMA_SxM1AR) (x = 0..7) 332
10.5.10 DMA stream x FIFO control register (DMA_SxFCR) (x = 0..7) . . . 333
10.5.11 DMA register map . . . 335

11 Chrom-Art Accelerator™ controller (DMA2D) . . . 339
11.1 DMA2D introduction . . . 339
11.2 DMA2D main features . . . 340
11.3 DMA2D functional description . . . 340
11.3.1 General description . . . 340
11.3.2 DMA2D control . . . 341
11.3.3 DMA2D foreground and background FIFOs . . . 341
11.3.4 DMA2D foreground and background pixel format converter (PFC) . . . 342
11.3.5 DMA2D foreground and background CLUT interface . . . 344
11.3.6 DMA2D blender . . . 345
11.3.7 DMA2D output PFC . . . 345
11.3.8 DMA2D output FIFO . . . 346
11.3.9 DMA2D AHB master port timer . . . 346
11.3.10 DMA2D transactions . . . 347
11.3.11 DMA2D configuration . . . 347
11.3.12 DMA2D transfer control (start, suspend, abort and completion) . . . 350
11.3.13 Watermark . . . 351
11.3.14 Error management . . . 351
11.3.15 AHB dead time . . . 351
11.4 DMA2D interrupts . . . 351
11.5 DMA2D registers . . . 352

11.5.1 DMA2D control register (DMA2D_CR) . . . 352
11.5.2 DMA2D Interrupt Status Register (DMA2D_ISR) . . . 354
11.5.3 DMA2D interrupt flag clear register (DMA2D_IFCR) . . . 355
11.5.4 DMA2D foreground memory address register (DMA2D_FGMAR) . . . 356
11.5.5 DMA2D foreground offset register (DMA2D_FGOR) . . . 356
11.5.6 DMA2D background memory address register (DMA2D_BGMAR) . . 357
11.5.7 DMA2D background offset register (DMA2D_BGOR) . . . 357
11.5.8 DMA2D foreground PFC control register (DMA2D_FGPFCCR) . . . 358
11.5.9 DMA2D foreground color register (DMA2D_FGCOLR) . . . 360
11.5.10 DMA2D background PFC control register (DMA2D_BGPFCCR) . . . 361
11.5.11 DMA2D background color register (DMA2D_BGCOLR) . . . 363
11.5.12 DMA2D foreground CLUT memory address register (DMA2D_FGCMAR) . . . 363
11.5.13 DMA2D background CLUT memory address register (DMA2D_BGCMAR) . . . 364
11.5.14 DMA2D output PFC control register (DMA2D_OPFCCR) . . . 364
11.5.15 DMA2D output color register (DMA2D_OCOLR) . . . 365
11.5.16 DMA2D output memory address register (DMA2D_OMAR) . . . 366
11.5.17 DMA2D output offset register (DMA2D_OOR) . . . 367
11.5.18 DMA2D number of line register (DMA2D_NLR) . . . 367
11.5.19 DMA2D line watermark register (DMA2D_LWR) . . . 368
11.5.20 DMA2D AHB master timer configuration register (DMA2D_AMTCR) . 368
11.5.21 DMA2D register map . . . 369

12 Interrupts and events . . . 371

12.1 Nested vectored interrupt controller (NVIC) . . . 371
12.1.1 NVIC features . . . 371
12.1.2 SysTick calibration value register . . . 371
12.1.3 Interrupt and exception vectors . . . 371
12.2 External interrupt/event controller (EXTI) . . . 371
12.2.1 EXTI main features . . . 379
12.2.2 EXTI block diagram . . . 380
12.2.3 Wakeup event management . . . 380
12.2.4 Functional description . . . 380
12.2.5 External interrupt/event line mapping . . . 382
12.3 EXTI registers . . . 384
12.3.1 Interrupt mask register (EXTI_IMR) . . . 384

12.3.2 Event mask register (EXTI_EMR) . . . 384
12.3.3 Rising trigger selection register (EXTI_RTSR) . . . 385
12.3.4 Falling trigger selection register (EXTI_FTSR) . . . 385
12.3.5 Software interrupt event register (EXTI_SWIER) . . . 386
12.3.6 Pending register (EXTI_PR) . . . 386
12.3.7 EXTI register map . . . 387

13 Analog-to-digital converter (ADC) . . . 388
13.1 ADC introduction . . . 388
13.2 ADC main features . . . 388
13.3 ADC functional description . . . 389
13.3.1 ADC on-off control . . . 390
13.3.2 ADC clock . . . 390
13.3.3 Channel selection . . . 390
13.3.4 Single conversion mode . . . 391
13.3.5 Continuous conversion mode . . . 392
13.3.6 Timing diagram . . . 392
13.3.7 Analog watchdog . . . 392
13.3.8 Scan mode . . . 393
13.3.9 Injected channel management . . . 394
13.3.10 Discontinuous mode . . . 395
13.4 Data alignment . . . 396
13.5 Channel-wise programmable sampling time . . . 397
13.6 Conversion on external trigger and trigger polarity . . . 397
13.7 Fast conversion mode . . . 399
13.8 Data management . . . 400
13.8.1 Using the DMA . . . 400
13.8.2 Managing a sequence of conversions without using the DMA . . . 400
13.8.3 Conversions without DMA and without overrun detection . . . 401
13.9 Multi ADC mode . . . 401
13.9.1 Injected simultaneous mode . . . 404
13.9.2 Regular simultaneous mode . . . 405
13.9.3 Interleaved mode . . . 407
13.9.4 Alternate trigger mode . . . 408
13.9.5 Combined regular/injected simultaneous mode . . . 410
13.9.6 Combined regular simultaneous + alternate trigger mode . . . 411

13.10 Temperature sensor 412
13.11 Battery charge monitoring 414
13.12 ADC interrupts 414
13.13 ADC registers 415
13.13.1 ADC status register (ADC_SR) 415
13.13.2 ADC control register 1 (ADC_CR1) 416
13.13.3 ADC control register 2 (ADC_CR2) 418
13.13.4 ADC sample time register 1 (ADC_SMPR1) 420
13.13.5 ADC sample time register 2 (ADC_SMPR2) 420
13.13.6 ADC injected channel data offset register x (ADC_JOFRx) (x=1..4) . . 421
13.13.7 ADC watchdog higher threshold register (ADC_HTR) 421
13.13.8 ADC watchdog lower threshold register (ADC_LTR) 422
13.13.9 ADC regular sequence register 1 (ADC_SQR1) 422
13.13.10 ADC regular sequence register 2 (ADC_SQR2) 423
13.13.11 ADC regular sequence register 3 (ADC_SQR3) 423
13.13.12 ADC injected sequence register (ADC_JSQR) 424
13.13.13 ADC injected data register x (ADC_JDRx) (x= 1..4) 425
13.13.14 ADC regular data register (ADC_DR) 425
13.13.15 ADC Common status register (ADC_CSR) 426
13.13.16 ADC common control register (ADC_CCR) 427
13.13.17 ADC common regular data register for dual and triple modes (ADC_CDR) 430
13.13.18 ADC register map 430

14 Digital-to-analog converter (DAC) 433
14.1 DAC introduction 433
14.2 DAC main features 433
14.3 DAC functional description 435
14.3.1 DAC channel enable 435
14.3.2 DAC output buffer enable 435
14.3.3 DAC data format 435
14.3.4 DAC conversion 436
14.3.5 DAC output voltage 437
14.3.6 DAC trigger selection 437
14.3.7 DMA request 438
14.3.8 Noise generation 438
14.3.9 Triangle-wave generation 439

14.4 Dual DAC channel conversion 440
14.4.1 Independent trigger without wave generation 441
14.4.2 Independent trigger with single LFSR generation 441
14.4.3 Independent trigger with different LFSR generation 441
14.4.4 Independent trigger with single triangle generation 442
14.4.5 Independent trigger with different triangle generation 442
14.4.6 Simultaneous software start 442
14.4.7 Simultaneous trigger without wave generation 443
14.4.8 Simultaneous trigger with single LFSR generation 443
14.4.9 Simultaneous trigger with different LFSR generation 443
14.4.10 Simultaneous trigger with single triangle generation 444
14.4.11 Simultaneous trigger with different triangle generation 444
14.5 DAC registers 445
14.5.1 DAC control register (DAC_CR) 445
14.5.2 DAC software trigger register (DAC_SWTRIGR) 448
14.5.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1) 448
14.5.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1) 449
14.5.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1) 449
14.5.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2) 450
14.5.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2) 450
14.5.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2) 450
14.5.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD) 451
14.5.10 DUAL DAC 12-bit left aligned data holding register (DAC_DHR12LD) 451
14.5.11 DUAL DAC 8-bit right aligned data holding register (DAC_DHR8RD) 452
14.5.12 DAC channel1 data output register (DAC_DOR1) 452
14.5.13 DAC channel2 data output register (DAC_DOR2) 452
14.5.14 DAC status register (DAC_SR) 453
14.5.15 DAC register map 453

15 Digital camera interface (DCMI) 455

15.1 DCMI introduction . . . 455
15.2 DCMI main features . . . 455
15.3 DCMI pins . . . 455
15.4 DCMI clocks . . . 455
15.5 DCMI functional overview . . . 456
15.5.1 DMA interface . . . 457
15.5.2 DCMI physical interface . . . 457
15.5.3 Synchronization . . . 459
15.5.4 Capture modes . . . 462
15.5.5 Crop feature . . . 463
15.5.6 JPEG format . . . 465
15.5.7 FIFO . . . 465
15.6 Data format description . . . 465
15.6.1 Data formats . . . 465
15.6.2 Monochrome format . . . 466
15.6.3 RGB format . . . 466
15.6.4 YCbCr format . . . 466
15.7 DCMI interrupts . . . 467
15.8 DCMI register description . . . 467
15.8.1 DCMI control register 1 (DCMI_CR) . . . 467
15.8.2 DCMI status register (DCMI_SR) . . . 470
15.8.3 DCMI raw interrupt status register (DCMI_RIS) . . . 471
15.8.4 DCMI interrupt enable register (DCMI_IER) . . . 472
15.8.5 DCMI masked interrupt status register (DCMI_MIS) . . . 473
15.8.6 DCMI interrupt clear register (DCMI_ICR) . . . 474
15.8.7 DCMI embedded synchronization code register (DCMI_ESCR) . . . 475
15.8.8 DCMI embedded synchronization unmask register (DCMI_ESUR) . . . 476
15.8.9 DCMI crop window start (DCMI_CWSTRT) . . . 477
15.8.10 DCMI crop window size (DCMI_CWSIZE) . . . 477
15.8.11 DCMI data register (DCMI_DR) . . . 478
15.8.12 DCMI register map . . . 478

16 LCD-TFT controller (LTDC) . . . 480
16.1 Introduction . . . 480
16.2 LTDC main features . . . 480
16.3 LTDC functional description . . . 481

16.3.1 LTDC block diagram . . . 481
16.3.2 LTDC reset and clocks . . . 481
16.3.3 LCD-TFT pins and signal interface . . . 483
16.4 LTDC programmable parameters . . . 483
16.4.1 LTDC Global configuration parameters . . . 483
16.4.2 Layer programmable parameters . . . 486
16.5 LTDC interrupts . . . 490
16.6 LTDC programming procedure . . . 492
16.7 LTDC registers . . . 493
16.7.1 LTDC Synchronization Size Configuration Register (LTDC_SSCR) . . . 493
16.7.2 LTDC Back Porch Configuration Register (LTDC_BPCR) . . . 493
16.7.3 LTDC Active Width Configuration Register (LTDC_AWCR) . . . 494
16.7.4 LTDC Total Width Configuration Register (LTDC_TWCR) . . . 495
16.7.5 LTDC Global Control Register (LTDC_GCR) . . . 495
16.7.6 LTDC Shadow Reload Configuration Register (LTDC_SRCR) . . . 497
16.7.7 LTDC Background Color Configuration Register (LTDC_BCCR) . . . 497
16.7.8 LTDC Interrupt Enable Register (LTDC_IER) . . . 498
16.7.9 LTDC Interrupt Status Register (LTDC_ISR) . . . 499
16.7.10 LTDC Interrupt Clear Register (LTDC_ICR) . . . 499
16.7.11 LTDC Line Interrupt Position Configuration Register (LTDC_LIPCR) . 500
16.7.12 LTDC Current Position Status Register (LTDC_CPSR) . . . 500
16.7.13 LTDC Current Display Status Register (LTDC_CDSR) . . . 501
16.7.14 LTDC Layerx Control Register (LTDC_LxCR) (where x=1..2) . . . 502
16.7.15 LTDC Layerx Window Horizontal Position Configuration Register (LTDC_LxWHPCR) (where x=1..2) . . . 503
16.7.16 LTDC Layerx Window Vertical Position Configuration Register (LTDC_LxWVPCR) (where x=1..2) . . . 504
16.7.17 LTDC Layerx Color Keying Configuration Register (LTDC_LxCKCR) (where x=1..2) . . . 505
16.7.18 LTDC Layerx Pixel Format Configuration Register (LTDC_LxPFCR) (where x=1..2) . . . 505
16.7.19 LTDC Layerx Constant Alpha Configuration Register (LTDC_LxCACR) (where x=1..2) . . . 506
16.7.20 LTDC Layerx Default Color Configuration Register (LTDC_LxDCCR) (where x=1..2) . . . 506
16.7.21 LTDC Layerx Blending Factors Configuration Register (LTDC_LxBFCR) (where x=1..2) . . . 508
16.7.22 LTDC Layerx Color Frame Buffer Address Register (LTDC_LxCFBAR) (where x=1..2) . . . 509

16.7.23 LTDC Layerx Color Frame Buffer Length Register (LTDC_LxCFBLR) (where x=1..2) . . . 509
16.7.24 LTDC Layerx ColorFrame Buffer Line Number Register (LTDC_LxCFBLNR) (where x=1..2) . . . 510
16.7.25 LTDC Layerx CLUT Write Register (LTDC_LxCLUTWR) (where x=1..2) . . . 511
16.7.26 LTDC register map . . . 512

17 Advanced-control timers (TIM1 and TIM8) . . . 515

17.1 TIM1 and TIM8 introduction . . . 515
17.2 TIM1 and TIM8 main features . . . 516
17.3 TIM1 and TIM8 functional description . . . 518
17.3.1 Time-base unit . . . 518
17.3.2 Counter modes . . . 520
17.3.3 Repetition counter . . . 529
17.3.4 Clock selection . . . 532
17.3.5 Capture/compare channels . . . 535
17.3.6 Input capture mode . . . 538
17.3.7 PWM input mode . . . 539
17.3.8 Forced output mode . . . 539
17.3.9 Output compare mode . . . 540
17.3.10 PWM mode . . . 541
17.3.11 Complementary outputs and dead-time insertion . . . 544
17.3.12 Using the break function . . . 546
17.3.13 Clearing the OCxREF signal on an external event . . . 549
17.3.14 6-step PWM generation . . . 550
17.3.15 One-pulse mode . . . 551
17.3.16 Encoder interface mode . . . 552
17.3.17 Timer input XOR function . . . 555
17.3.18 Interfacing with Hall sensors . . . 555
17.3.19 TIMx and external trigger synchronization . . . 557
17.3.20 Timer synchronization . . . 560
17.3.21 Debug mode . . . 560
17.4 TIM1 and TIM8 registers . . . 561
17.4.1 TIM1 and TIM8 control register 1 (TIMx_CR1) . . . 561
17.4.2 TIM1 and TIM8 control register 2 (TIMx_CR2) . . . 562
17.4.3 TIM1 and TIM8 slave mode control register (TIMx_SMCR) . . . 565
17.4.4 TIM1 and TIM8 DMA/interrupt enable register (TIMx_DIER) . . . 567

17.4.5 TIM1 and TIM8 status register (TIMx_SR) 569
17.4.6 TIM1 and TIM8 event generation register (TIMx_EGR) 570
17.4.7 TIM1 and TIM8 capture/compare mode register 1 (TIMx_CCMR1) . . 572
17.4.8 TIM1 and TIM8 capture/compare mode register 2 (TIMx_CCMR2) . . 575
17.4.9 TIM1 and TIM8 capture/compare enable register (TIMx_CCER) 576
17.4.10 TIM1 and TIM8 counter (TIMx_CNT) 580
17.4.11 TIM1 and TIM8 prescaler (TIMx_PSC) 580
17.4.12 TIM1 and TIM8 auto-reload register (TIMx_ARR) 580
17.4.13 TIM1 and TIM8 repetition counter register (TIMx_RCR) 581
17.4.14 TIM1 and TIM8 capture/compare register 1 (TIMx_CCR1) 581
17.4.15 TIM1 and TIM8 capture/compare register 2 (TIMx_CCR2) 582
17.4.16 TIM1 and TIM8 capture/compare register 3 (TIMx_CCR3) 582
17.4.17 TIM1 and TIM8 capture/compare register 4 (TIMx_CCR4) 583
17.4.18 TIM1 and TIM8 break and dead-time register (TIMx_BDTR) 583
17.4.19 TIM1 and TIM8 DMA control register (TIMx_DCR) 585
17.4.20 TIM1 and TIM8 DMA address for full transfer (TIMx_DMAR) 586
17.4.21 TIM1 and TIM8 register map 587

18 General-purpose timers (TIM2 to TIM5) 589

18.1 TIM2 to TIM5 introduction 589
18.2 TIM2 to TIM5 main features 589
18.3 TIM2 to TIM5 functional description 591
18.3.1 Time-base unit 591
18.3.2 Counter modes 592
18.3.3 Clock selection 602
18.3.4 Capture/compare channels 605
18.3.5 Input capture mode 607
18.3.6 PWM input mode 608
18.3.7 Forced output mode 609
18.3.8 Output compare mode 609
18.3.9 PWM mode 610
18.3.10 One-pulse mode 613
18.3.11 Clearing the OCxREF signal on an external event 614
18.3.12 Encoder interface mode 615
18.3.13 Timer input XOR function 618
18.3.14 Timers and external trigger synchronization 618
18.3.15 Timer synchronization 621

18.3.16 Debug mode 626
18.4 TIM2 to TIM5 registers 627
18.4.1 TIMx control register 1 (TIMx_CR1) 627
18.4.2 TIMx control register 2 (TIMx_CR2) 629
18.4.3 TIMx slave mode control register (TIMx_SMCR) 630
18.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER) 632
18.4.5 TIMx status register (TIMx_SR) 633
18.4.6 TIMx event generation register (TIMx_EGR) 635
18.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1) 636
18.4.8 TIMx capture/compare mode register 2 (TIMx_CCMR2) 639
18.4.9 TIMx capture/compare enable register (TIMx_CCER) 640
18.4.10 TIMx counter (TIMx_CNT) 642
18.4.11 TIMx prescaler (TIMx_PSC) 642
18.4.12 TIMx auto-reload register (TIMx_ARR) 642
18.4.13 TIMx capture/compare register 1 (TIMx_CCR1) 643
18.4.14 TIMx capture/compare register 2 (TIMx_CCR2) 643
18.4.15 TIMx capture/compare register 3 (TIMx_CCR3) 644
18.4.16 TIMx capture/compare register 4 (TIMx_CCR4) 644
18.4.17 TIMx DMA control register (TIMx_DCR) 645
18.4.18 TIMx DMA address for full transfer (TIMx_DMAR) 646
18.4.19 TIM2 option register (TIM2_OR) 647
18.4.20 TIM5 option register (TIM5_OR) 647
18.4.21 TIMx register map 648

19 General-purpose timers (TIM9 to TIM14) 650
19.1 TIM9 to TIM14 introduction 650
19.2 TIM9 to TIM14 main features 650
19.2.1 TIM9/TIM12 main features 650
19.2.2 TIM10/TIM11 and TIM13/TIM14 main features 651
19.3 TIM9 to TIM14 functional description 653
19.3.1 Time-base unit 653
19.3.2 Counter modes 655
19.3.3 Clock selection 658
19.3.4 Capture/compare channels 660
19.3.5 Input capture mode 661
19.3.6 PWM input mode (only for TIM9/12) 663
19.3.7 Forced output mode 664

19.3.8 Output compare mode 664
19.3.9 PWM mode 665
19.3.10 One-pulse mode 666
19.3.11 TIM9/12 external trigger synchronization 668
19.3.12 Timer synchronization (TIM9/12) 671
19.3.13 Debug mode 671
19.4 TIM9 and TIM12 registers 672
19.4.1 TIM9/12 control register 1 (TIMx_CR1) 672
19.4.2 TIM9/12 slave mode control register (TIMx_SMCR) 673
19.4.3 TIM9/12 Interrupt enable register (TIMx_DIER) 674
19.4.4 TIM9/12 status register (TIMx_SR) 676
19.4.5 TIM9/12 event generation register (TIMx_EGR) 677
19.4.6 TIM9/12 capture/compare mode register 1 (TIMx_CCMR1) 678
19.4.7 TIM9/12 capture/compare enable register (TIMx_CCER) 681
19.4.8 TIM9/12 counter (TIMx_CNT) 682
19.4.9 TIM9/12 prescaler (TIMx_PSC) 682
19.4.10 TIM9/12 auto-reload register (TIMx_ARR) 682
19.4.11 TIM9/12 capture/compare register 1 (TIMx_CCR1) 683
19.4.12 TIM9/12 capture/compare register 2 (TIMx_CCR2) 683
19.4.13 TIM9/12 register map 684
19.5 TIM10/11/13/14 registers 686
19.5.1 TIM10/11/13/14 control register 1 (TIMx_CR1) 686
19.5.2 TIM10/11/13/14 Interrupt enable register (TIMx_DIER) 687
19.5.3 TIM10/11/13/14 status register (TIMx_SR) 687
19.5.4 TIM10/11/13/14 event generation register (TIMx_EGR) 688
19.5.5 TIM10/11/13/14 capture/compare mode register 1 (TIMx_CCMR1) . . 688
19.5.6 TIM10/11/13/14 capture/compare enable register (TIMx_CCER) 691
19.5.7 TIM10/11/13/14 counter (TIMx_CNT) 692
19.5.8 TIM10/11/13/14 prescaler (TIMx_PSC) 692
19.5.9 TIM10/11/13/14 auto-reload register (TIMx_ARR) 692
19.5.10 TIM10/11/13/14 capture/compare register 1 (TIMx_CCR1) 693
19.5.11 TIM11 option register 1 (TIM11_OR) 693
19.5.12 TIM10/11/13/14 register map 694

20 Basic timers (TIM6 and TIM7) 696
20.1 TIM6 and TIM7 introduction 696
20.2 TIM6 and TIM7 main features 696

20.3 TIM6 and TIM7 functional description 697
20.3.1 Time-base unit 697
20.3.2 Counting mode 699
20.3.3 Clock source 701
20.3.4 Debug mode 702
20.4 TIM6 and TIM7 registers 702
20.4.1 TIM6 and TIM7 control register 1 (TIMx_CR1) 702
20.4.2 TIM6 and TIM7 control register 2 (TIMx_CR2) 704
20.4.3 TIM6 and TIM7 DMA/Interrupt enable register (TIMx_DIER) 704
20.4.4 TIM6 and TIM7 status register (TIMx_SR) 705
20.4.5 TIM6 and TIM7 event generation register (TIMx_EGR) 705
20.4.6 TIM6 and TIM7 counter (TIMx_CNT) 705
20.4.7 TIM6 and TIM7 prescaler (TIMx_PSC) 706
20.4.8 TIM6 and TIM7 auto-reload register (TIMx_ARR) 706
20.4.9 TIM6 and TIM7 register map 707

21 Independent watchdog (IWDG) 708
21.1 IWDG introduction 708
21.2 IWDG main features 708
21.3 IWDG functional description 708
21.3.1 Hardware watchdog 708
21.3.2 Register access protection 708
21.3.3 Debug mode 709
21.4 IWDG registers 710
21.4.1 Key register (IWDG_KR) 710
21.4.2 Prescaler register (IWDG_PR) 710
21.4.3 Reload register (IWDG_RLR) 711
21.4.4 Status register (IWDG_SR) 711
21.4.5 IWDG register map 712

22 Window watchdog (WWDG) 713
22.1 WWDG introduction 713
22.2 WWDG main features 713
22.3 WWDG functional description 713
22.4 How to program the watchdog timeout 715
22.5 Debug mode 716

22.6 WWDG registers 717
22.6.1 Control register (WWDG_CR) 717
22.6.2 Configuration register (WWDG_CFR) 718
22.6.3 Status register (WWDG_SR) 718
22.6.4 WWDG register map 719

23 Cryptographic processor (CRYP) 720
23.1 CRYP introduction 720
23.2 CRYP main features 720
23.3 CRYP functional description 722
23.3.1 DES/TDES cryptographic core 723
23.3.2 AES cryptographic core 728
23.3.3 Data type 739
23.3.4 Initialization vectors - CRYP_IV0...1(L/R) 742
23.3.5 CRYP busy state 743
23.3.6 Procedure to perform an encryption or a decryption 744
23.3.7 Context swapping 745
23.4 CRYP interrupts 747
23.5 CRYP DMA interface 748
23.6 CRYP registers 748
23.6.1 CRYP control register (CRYP_CR) for STM32F415/417xx 748
23.6.2 CRYP control register (CRYP_CR) for STM32F415/417xx 750
23.6.3 CRYP status register (CRYP_SR) 753
23.6.4 CRYP data input register (CRYP_DIN) 754
23.6.5 CRYP data output register (CRYP_DOUT) 755
23.6.6 CRYP DMA control register (CRYP_DMACR) 756
23.6.7 CRYP interrupt mask set/clear register (CRYP_IMSCR) 756
23.6.8 CRYP raw interrupt status register (CRYP_RISR) 757
23.6.9 CRYP masked interrupt status register (CRYP_MISR) 757
23.6.10 CRYP key registers (CRYP_K0...3(L/R)R) 758
23.6.11 CRYP initialization vector registers (CRYP_IV0...1(L/R)R) 760
23.6.12 CRYP context swap registers (CRYP_CSGCMCCM0..7R and CRYP_CSGCM0..7R) for STM32F42xxx and STM32F43xxx 762
23.6.13 CRYP register map 763

24 Random number generator (RNG) 767
24.1 RNG introduction 767

24.2 RNG main features . . . 767
24.3 RNG functional description . . . 767
24.3.1 Operation . . . 768
24.3.2 Error management . . . 768
24.4 RNG registers . . . 768
24.4.1 RNG control register (RNG_CR) . . . 769
24.4.2 RNG status register (RNG_SR) . . . 769
24.4.3 RNG data register (RNG_DR) . . . 770
24.4.4 RNG register map . . . 771

25 Hash processor (HASH) . . . 772
25.1 HASH introduction . . . 772
25.2 HASH main features . . . 772
25.3 HASH functional description . . . 773
25.3.1 Duration of the processing . . . 775
25.3.2 Data type . . . 775
25.3.3 Message digest computing . . . 777
25.3.4 Message padding . . . 778
25.3.5 Hash operation . . . 779
25.3.6 HMAC operation . . . 779
25.3.7 Context swapping . . . 780
25.3.8 HASH interrupt . . . 782
25.4 HASH registers . . . 782
25.4.1 HASH control register (HASH_CR) for STM32F415/417xx . . . 782
25.4.2 HASH control register (HASH_CR) for STM32F43xxx . . . 785
25.4.3 HASH data input register (HASH_DIN) . . . 788
25.4.4 HASH start register (HASH_STR) . . . 789
25.4.5 HASH digest registers (HASH_HR0..4/5/6/7) . . . 790
25.4.6 HASH interrupt enable register (HASH_IMR) . . . 792
25.4.7 HASH status register (HASH_SR) . . . 793
25.4.8 HASH context swap registers (HASH_CSRx) . . . 794
25.4.9 HASH register map . . . 795

26 Real-time clock (RTC) . . . 798
26.1 Introduction . . . 798
26.2 RTC main features . . . 799

26.3 RTC functional description 801
26.3.1 Clock and prescalers 801
26.3.2 Real-time clock and calendar 802
26.3.3 Programmable alarms 802
26.3.4 Periodic auto-wakeup 802
26.3.5 RTC initialization and configuration 803
26.3.6 Reading the calendar 805
26.3.7 Resetting the RTC 806
26.3.8 RTC synchronization 806
26.3.9 RTC reference clock detection 807
26.3.10 RTC coarse digital calibration 808
26.3.11 RTC smooth digital calibration 809
26.3.12 Timestamp function 811
26.3.13 Tamper detection 811
26.3.14 Calibration clock output 813
26.3.15 Alarm output 813
26.4 RTC and low-power modes 814
26.5 RTC interrupts 814
26.6 RTC registers 816
26.6.1 RTC time register (RTC_TR) 816
26.6.2 RTC date register (RTC_DR) 817
26.6.3 RTC control register (RTC_CR) 818
26.6.4 RTC initialization and status register (RTC_ISR) 820
26.6.5 RTC prescaler register (RTC_PRER) 823
26.6.6 RTC wakeup timer register (RTC_WUTR) 823
26.6.7 RTC calibration register (RTC_CALIBR) 824
26.6.8 RTC alarm A register (RTC_ALRMAR) 825
26.6.9 RTC alarm B register (RTC_ALRMBR) 826
26.6.10 RTC write protection register (RTC_WPR) 827
26.6.11 RTC sub second register (RTC_SSR) 827
26.6.12 RTC shift control register (RTC_SHIFTR) 828
26.6.13 RTC time stamp time register (RTC_TSTR) 828
26.6.14 RTC time stamp date register (RTC_TSDR) 829
26.6.15 RTC timestamp sub second register (RTC_TSSSR) 830
26.6.16 RTC calibration register (RTC_CALR) 830
26.6.17 RTC tamper and alternate function configuration register (RTC_TAFCR) 832

26.6.18 RTC alarm A sub second register (RTC_ALRMASSR) 834
26.6.19 RTC alarm B sub second register (RTC_ALRMBSSR) 835
26.6.20 RTC backup registers (RTC_BKPxR) 836
26.6.21 RTC register map 836

27 Inter-integrated circuit (I2C) interface 839
27.1 I²C introduction 839
27.2 I²C main features 839
27.3 I²C functional description 840
27.3.1 Mode selection 840
27.3.2 I2C slave mode 843
27.3.3 I2C master mode 846
27.3.4 Error conditions 851
27.3.5 Programmable noise filter 852
27.3.6 SDA/SCL line control 853
27.3.7 SMBus 853
27.3.8 DMA requests 856
27.3.9 Packet error checking 857
27.4 I²C interrupts 858
27.5 I²C debug mode 860
27.6 I²C registers 860
27.6.1 I²C Control register 1 (I2C_CR1) 860
27.6.2 I²C Control register 2 (I2C_CR2) 862
27.6.3 I²C Own address register 1 (I2C_OAR1) 864
27.6.4 I²C Own address register 2 (I2C_OAR2) 864
27.6.5 I²C Data register (I2C_DR) 865
27.6.6 I²C Status register 1 (I2C_SR1) 865
27.6.7 I²C Status register 2 (I2C_SR2) 868
27.6.8 I²C Clock control register (I2C_CCR) 870
27.6.9 I²C TRISE register (I2C_TRISE) 871
27.6.10 I²C FLTR register (I2C_FLTR) 871
27.6.11 I2C register map 872

28 Serial peripheral interface (SPI) 873
28.1 SPI introduction 873
28.2 SPI and I²S main features 874

28.2.1 SPI features . . . 874
28.2.2 I2S features . . . 875
28.3 SPI functional description . . . 876
28.3.1 General description . . . 876
28.3.2 Configuring the SPI in slave mode . . . 880
28.3.3 Configuring the SPI in master mode . . . 882
28.3.4 Configuring the SPI for half-duplex communication . . . 884
28.3.5 Data transmission and reception procedures . . . 885
28.3.6 CRC calculation . . . 891
28.3.7 Status flags . . . 893
28.3.8 Disabling the SPI . . . 894
28.3.9 SPI communication using DMA (direct memory addressing) . . . 895
28.3.10 Error flags . . . 897
28.3.11 SPI interrupts . . . 898
28.4 I2S functional description . . . 899
28.4.1 I2S general description . . . 899
28.4.2 I2S full duplex . . . 900
28.4.3 Supported audio protocols . . . 901
28.4.4 Clock generator . . . 907
28.4.5 I2S master mode . . . 909
28.4.6 I2S slave mode . . . 911
28.4.7 Status flags . . . 913
28.4.8 Error flags . . . 914
28.4.9 I2S interrupts . . . 915
28.4.10 DMA features . . . 915
28.5 SPI and I2S registers . . . 916
28.5.1 SPI control register 1 (SPI_CR1) (not used in I2S mode) . . . 916
28.5.2 SPI control register 2 (SPI_CR2) . . . 918
28.5.3 SPI status register (SPI_SR) . . . 919
28.5.4 SPI data register (SPI_DR) . . . 920
28.5.5 SPI CRC polynomial register (SPI_CRCPR) (not used in I2S mode) . . . 921
28.5.6 SPI RX CRC register (SPI_RXCRCR) (not used in I2S mode) . . . 921
28.5.7 SPI TX CRC register (SPI_TXCRCR) (not used in I2S mode) . . . 922
28.5.8 SPI_I2S configuration register (SPI_I2SCFGR) . . . 922
28.5.9 SPI_I2S prescaler register (SPI_I2SPR) . . . 923
28.5.10 SPI register map . . . 925

29 Serial audio interface (SAI) 926
29.1 Introduction 926
29.2 Main features 927
29.3 Functional block diagram 928
29.4 Main SAI modes 929
29.5 SAI synchronization mode 930
29.6 Audio data size 930
29.7 Frame synchronization 930
29.7.1 Frame length 931
29.7.2 Frame synchronization polarity 931
29.7.3 Frame synchronization active level length 932
29.7.4 Frame synchronization offset 932
29.7.5 FS signal role 932
29.8 Slot configuration 933
29.9 SAI clock generator 935
29.10 Internal FIFOs 936
29.11 AC’97 link controller 939
29.12 Specific features 940
29.12.1 Mute mode 940
29.12.2 MONO/STEREO function 940
29.12.3 Companding mode 941
29.12.4 Output data line management on an inactive slot 942
29.13 Error flags 944
29.13.1 FIFO overrun/underrun (OVRUDR) 944
29.13.2 Anticipated frame synchronisation detection (AFSDET) 946
29.13.3 Late frame synchronization detection 946
29.13.4 Codec not ready (CNRDY AC’97) 947
29.13.5 Wrong clock configuration in master mode (with NODIV = 0) 947
29.14 Interrupt sources 948
29.15 Disabling the SAI 948
29.16 SAI DMA interface 949
29.17 SAI registers 950
29.17.1 SAI xConfiguration register 1 (SAI_xCR1) where x is A or B 950
29.17.2 SAI xConfiguration register 2 (SAI_xCR2) where x is A or B 953
29.17.3 SAI xFrame configuration register (SAI_XFRCR) where x is A or B . . 955

29.17.4 SAI xSlot register (SAI_xSLOTR) where x is A or B 957
29.17.5 SAI xInterrupt mask register (SAI_xIM) where x is A or B 958
29.17.6 SAI xStatus register (SAI_xSR) where x is A or B 960
29.17.7 SAI xClear flag register (SAI_xCLRFR) where X is A or B 962
29.17.8 SAI xData register (SAI_xDR) where x is A or B 963
29.17.9 SAI register map 963

30 Universal synchronous asynchronous receiver transmitter (USART) 965

30.1 USART introduction 965
30.2 USART main features 965
30.3 USART functional description 966
30.3.1 USART character description 969
30.3.2 Transmitter 970
30.3.3 Receiver 973
30.3.4 Fractional baud rate generation 978
30.3.5 USART receiver tolerance to clock deviation 988
30.3.6 Multiprocessor communication 989
30.3.7 Parity control 991
30.3.8 LIN (local interconnection network) mode 992
30.3.9 USART synchronous mode 994
30.3.10 Single-wire half-duplex communication 996
30.3.11 Smartcard 997
30.3.12 IrDA SIR ENDEC block 999
30.3.13 Continuous communication using DMA 1001
30.3.14 Hardware flow control 1003
30.4 USART interrupts 1006
30.5 USART mode configuration 1007
30.6 USART registers 1007
30.6.1 Status register (USART_SR) 1007
30.6.2 Data register (USART_DR) 1010
30.6.3 Baud rate register (USART_BRR) 1010
30.6.4 Control register 1 (USART_CR1) 1010
30.6.5 Control register 2 (USART_CR2) 1013
30.6.6 Control register 3 (USART_CR3) 1014
30.6.7 Guard time and prescaler register (USART_GTPR) 1017
30.6.8 USART register map 1018

31 Secure digital input/output interface (SDIO) 1019
31.1 SDIO main features 1019
31.2 SDIO bus topology 1020
31.3 SDIO functional description 1022
31.3.1 SDIO adapter 1023
31.3.2 SDIO APB2 interface 1033
31.4 Card functional description 1034
31.4.1 Card identification mode 1034
31.4.2 Card reset 1034
31.4.3 Operating voltage range validation 1034
31.4.4 Card identification process 1035
31.4.5 Block write 1036
31.4.6 Block read 1037
31.4.7 Stream access, stream write and stream read (MultiMediaCard only) 1037
31.4.8 Erase: group erase and sector erase 1039
31.4.9 Wide bus selection or deselection 1039
31.4.10 Protection management 1039
31.4.11 Card status register 1042
31.4.12 SD status register 1045
31.4.13 SD I/O mode 1049
31.4.14 Commands and responses 1050
31.5 Response formats 1054
31.5.1 R1 (normal response command) 1054
31.5.2 R1b 1054
31.5.3 R2 (CID, CSD register) 1054
31.5.4 R3 (OCR register) 1055
31.5.5 R4 (Fast I/O) 1055
31.5.6 R4b 1056
31.5.7 R5 (interrupt request) 1056
31.5.8 R6 1057
31.6 SDIO I/O card-specific operations 1057
31.6.1 SDIO I/O read wait operation by SDIO_D2 signaling 1058
31.6.2 SDIO read wait operation by stopping SDIO_CK 1058
31.6.3 SDIO suspend/resume operation 1058
31.6.4 SDIO interrupts 1058

31.7 CE-ATA specific operations 1059
31.7.1 Command completion signal disable 1059
31.7.2 Command completion signal enable 1059
31.7.3 CE-ATA interrupt 1059
31.7.4 Aborting CMD61 1059
31.8 HW flow control 1060
31.9 SDIO registers 1060
31.9.1 SDIO power control register (SDIO_POWER) 1060
31.9.2 SDI clock control register (SDIO_CLKCR) 1061
31.9.3 SDIO argument register (SDIO_ARG) 1062
31.9.4 SDIO command register (SDIO_CMD) 1062
31.9.5 SDIO command response register (SDIO_RESPCMD) 1063
31.9.6 SDIO response 1..4 register (SDIO_RESPx) 1064
31.9.7 SDIO data timer register (SDIO_DTIMER) 1064
31.9.8 SDIO data length register (SDIO_DLEN) 1065
31.9.9 SDIO data control register (SDIO_DCTRL) 1066
31.9.10 SDIO data counter register (SDIO_DCOUNT) 1067
31.9.11 SDIO status register (SDIO_STA) 1068
31.9.12 SDIO interrupt clear register (SDIO_ICR) 1069
31.9.13 SDIO mask register (SDIO_MASK) 1071
31.9.14 SDIO FIFO counter register (SDIO_FIFOCNT) 1073
31.9.15 SDIO data FIFO register (SDIO_FIFO) 1074
31.9.16 SDIO register map 1074

32 Controller area network (bxCAN) 1076
32.1 bxCAN introduction 1076
32.2 bxCAN main features 1076
32.3 bxCAN general description 1077
32.3.1 CAN 2.0B active core 1077
32.3.2 Control, status and configuration registers 1078
32.3.3 Tx mailboxes 1078
32.3.4 Acceptance filters 1078
32.4 bxCAN operating modes 1079
32.4.1 Initialization mode 1080
32.4.2 Normal mode 1080
32.4.3 Sleep mode (low power) 1080

32.5 Test mode 1081
32.5.1 Silent mode 1081
32.5.2 Loop back mode 1082
32.5.3 Loop back combined with silent mode 1082
32.6 Debug mode 1083
32.7 bxCAN functional description 1083
32.7.1 Transmission handling 1083
32.7.2 Time triggered communication mode 1085
32.7.3 Reception handling 1085
32.7.4 Identifier filtering 1087
32.7.5 Message storage 1091
32.7.6 Error management 1093
32.7.7 Bit timing 1093
32.8 bxCAN interrupts 1095
32.9 CAN registers 1097
32.9.1 Register access protection 1097
32.9.2 CAN control and status registers 1097
32.9.3 CAN mailbox registers 1107
32.9.4 CAN filter registers 1114
32.9.5 bxCAN register map 1118

33 Ethernet (ETH): media access control (MAC) with DMA controller 1122

33.1 Ethernet introduction 1122
33.2 Ethernet main features 1122
33.2.1 MAC core features 1123
33.2.2 DMA features 1124
33.2.3 PTP features 1124
33.3 Ethernet pins 1125
33.4 Ethernet functional description: SMI, MII and RMII 1126
33.4.1 Station management interface: SMI 1126
33.4.2 Media-independent interface: MII 1129
33.4.3 Reduced media-independent interface: RMII 1131
33.4.4 MII/RMII selection 1132
33.5 Ethernet functional description: MAC 802.3 1133
33.5.1 MAC 802.3 frame format 1134

33.5.2 MAC frame transmission 1137
33.5.3 MAC frame reception 1144
33.5.4 MAC interrupts 1149
33.5.5 MAC filtering 1150
33.5.6 MAC loopback mode 1153
33.5.7 MAC management counters: MMC 1153
33.5.8 Power management: PMT 1154
33.5.9 Precision time protocol (IEEE1588 PTP) 1157
33.6 Ethernet functional description: DMA controller operation 1163
33.6.1 Initialization of a transfer using DMA 1164
33.6.2 Host bus burst access 1164
33.6.3 Host data buffer alignment 1165
33.6.4 Buffer size calculations 1165
33.6.5 DMA arbiter 1166
33.6.6 Error response to DMA 1166
33.6.7 Tx DMA configuration 1166
33.6.8 Rx DMA configuration 1178
33.6.9 DMA interrupts 1189
33.7 Ethernet interrupts 1190
33.8 Ethernet register descriptions 1191
33.8.1 MAC register description 1191
33.8.2 MMC register description 1210
33.8.3 IEEE 1588 time stamp registers 1215
33.8.4 DMA register description 1223
33.8.5 Ethernet register maps 1236

34 USB on-the-go full-speed (OTG_FS) 1240
34.1 OTG_FS introduction 1240
34.2 OTG_FS main features 1241
34.2.1 General features 1241
34.2.2 Host-mode features 1242
34.2.3 Peripheral-mode features 1242
34.3 OTG_FS functional description 1243
34.3.1 OTG pins 1243
34.3.2 OTG full-speed core 1243
34.3.3 Full-speed OTG PHY 1244

34.4 OTG dual role device (DRD) . . . 1245
34.4.1 ID line detection . . . 1245
34.4.2 HNP dual role device . . . 1245
34.4.3 SRP dual role device . . . 1246
34.5 USB peripheral . . . 1246
34.5.1 SRP-capable peripheral . . . 1247
34.5.2 Peripheral states . . . 1247
34.5.3 Peripheral endpoints . . . 1248
34.6 USB host . . . 1250
34.6.1 SRP-capable host . . . 1251
34.6.2 USB host states . . . 1251
34.6.3 Host channels . . . 1253
34.6.4 Host scheduler . . . 1254
34.7 SOF trigger . . . 1255
34.7.1 Host SOFs . . . 1255
34.7.2 Peripheral SOFs . . . 1256
34.8 OTG low-power modes . . . 1256
34.9 Dynamic update of the OTG_FS_HFIR register . . . 1257
34.10 USB data FIFOs . . . 1258
34.11 Peripheral FIFO architecture . . . 1259
34.11.1 Peripheral Rx FIFO . . . 1259
34.11.2 Peripheral Tx FIFOs . . . 1260
34.12 Host FIFO architecture . . . 1260
34.12.1 Host Rx FIFO . . . 1260
34.12.2 Host Tx FIFOs . . . 1261
34.13 FIFO RAM allocation . . . 1261
34.13.1 Device mode . . . 1261
34.13.2 Host mode . . . 1262
34.14 USB system performance . . . 1262
34.15 OTG_FS interrupts . . . 1263
34.16 OTG_FS control and status registers . . . 1265
34.16.1 CSR memory map . . . 1266
34.16.2 OTG_FS global registers . . . 1271
34.16.3 Host-mode registers . . . 1292
34.16.4 Device-mode registers . . . 1302

34.16.5 OTG_FS power and clock gating control register (OTG_FS_PCGCCTL) . . . 1325
34.16.6 OTG_FS register map . . . 1326
34.17 OTG_FS programming model . . . 1335
34.17.1 Core initialization . . . 1335
34.17.2 Host initialization . . . 1336
34.17.3 Device initialization . . . 1336
34.17.4 Host programming model . . . 1337
34.17.5 Device programming model . . . 1353
34.17.6 Operational model . . . 1355
34.17.7 Worst case response time . . . 1372
34.17.8 OTG programming model . . . 1374

35 USB on-the-go high-speed (OTG_HS) . . . 1381
35.1 OTG_HS introduction . . . 1381
35.2 OTG_HS main features . . . 1381
35.2.1 General features . . . 1382
35.2.2 Host-mode features . . . 1383
35.2.3 Peripheral-mode features . . . 1383
35.3 OTG_HS functional description . . . 1383
35.3.1 OTG pins . . . 1384
35.3.2 High-speed OTG PHY . . . 1384
35.3.3 Embedded Full-speed OTG PHY . . . 1385
35.4 OTG dual-role device . . . 1385
35.4.1 ID line detection . . . 1385
35.4.2 HNP dual role device . . . 1385
35.4.3 SRP dual-role device . . . 1386
35.5 USB functional description in peripheral mode . . . 1386
35.5.1 SRP-capable peripheral . . . 1386
35.5.2 Peripheral states . . . 1387
35.5.3 Peripheral endpoints . . . 1388
35.6 USB functional description on host mode . . . 1391
35.6.1 SRP-capable host . . . 1391
35.6.2 USB host states . . . 1392
35.6.3 Host channels . . . 1393
35.6.4 Host scheduler . . . 1395

35.7 SOF trigger . . . 1396
35.7.1 Host SOFs . . . 1396
35.7.2 Peripheral SOFs . . . 1396
35.8 OTG_HS low-power modes . . . 1397
35.9 Dynamic update of the OTG_HS_HFIR register . . . 1398
35.10 FIFO RAM allocation . . . 1399
35.10.1 Peripheral mode . . . 1399
35.10.2 Host mode . . . 1399
35.11 OTG_HS interrupts . . . 1400
35.12 OTG_HS control and status registers . . . 1402
35.12.1 CSR memory map . . . 1402
35.12.2 OTG_HS global registers . . . 1407
35.12.3 Host-mode registers . . . 1430
35.12.4 Device-mode registers . . . 1443
35.12.5 OTG_HS power and clock gating control register (OTG_HS_PCGCCTL) . . . 1472
35.12.6 OTG_HS register map . . . 1472
35.13 OTG_HS programming model . . . 1487
35.13.1 Core initialization . . . 1487
35.13.2 Host initialization . . . 1488
35.13.3 Device initialization . . . 1489
35.13.4 DMA mode . . . 1489
35.13.5 Host programming model . . . 1489
35.13.6 Device programming model . . . 1515
35.13.7 Operational model . . . 1517
35.13.8 Worst case response time . . . 1536
35.13.9 OTG programming model . . . 1538

36 Flexible static memory controller (FSMC) . . . 1544
36.1 FSMC main features . . . 1544
36.2 Block diagram . . . 1545
36.3 AHB interface . . . 1545
36.3.1 Supported memories and transactions . . . 1546
36.4 External device address mapping . . . 1547
36.4.1 NOR/PSRAM address mapping . . . 1547
36.4.2 NAND/PC Card address mapping . . . 1548

36.5 NOR Flash/PSRAM controller 1549
36.5.1 External memory interface signals 1550
36.5.2 Supported memories and transactions 1552
36.5.3 General timing rules 1553
36.5.4 NOR Flash/PSRAM controller asynchronous transactions 1554
36.5.5 Synchronous transactions 1571
36.5.6 NOR/PSRAM control registers 1577
36.6 NAND Flash/PC Card controller 1584
36.6.1 External memory interface signals 1585
36.6.2 NAND Flash / PC Card supported memories and transactions 1587
36.6.3 Timing diagrams for NAND and PC Card 1587
36.6.4 NAND Flash operations 1588
36.6.5 NAND Flash prewait functionality 1589
36.6.6 Computation of the error correction code (ECC) in NAND Flash memory 1590
36.6.7 PC Card/CompactFlash operations 1591
36.6.8 NAND Flash/PC Card control registers 1593
36.6.9 FSMC register map 1600

37 Flexible memory controller (FMC) 1602
37.1 FMC main features 1602
37.2 Block diagram 1603
37.3 AHB interface 1604
37.3.1 Supported memories and transactions 1605
37.4 External device address mapping 1606
37.4.1 NOR/PSRAM address mapping 1607
37.4.2 NAND Flash memory/PC Card address mapping 1608
37.4.3 SDRAM address mapping 1609
37.5 NOR Flash/PSRAM controller 1612
37.5.1 External memory interface signals 1613
37.5.2 Supported memories and transactions 1615
37.5.3 General timing rules 1617
37.5.4 NOR Flash/PSRAM controller asynchronous transactions 1617
37.5.5 Synchronous transactions 1634
37.5.6 NOR/PSRAM controller registers 1640
37.6 NAND Flash/PC Card controller 1647
37.6.1 External memory interface signals 1648

37.6.2 NAND Flash / PC Card supported memories and transactions 1650
37.6.3 Timing diagrams for NAND Flash memory and PC Card 1650
37.6.4 NAND Flash operations 1651
37.6.5 NAND Flash prewait functionality 1652
37.6.6 Computation of the error correction code (ECC) in NAND Flash memory 1653
37.6.7 PC Card/CompactFlash operations 1654
37.6.8 NAND Flash/PC Card controller registers 1656
37.7 SDRAM controller 1663
37.7.1 SDRAM controller main features 1663
37.7.2 SDRAM External memory interface signals 1663
37.7.3 SDRAM controller functional description 1664
37.7.4 Low power modes 1671
37.7.5 SDRAM controller registers 1674
37.8 FMC register map 1680

38 Debug support (DBG) 1683
38.1 Overview 1683
38.2 Reference Arm® documentation 1684
38.3 SWJ debug port (serial wire and JTAG) 1684
38.3.1 Mechanism to select the JTAG-DP or the SW-DP 1685
38.4 Pinout and debug port pins 1685
38.4.1 SWJ debug port pins 1686
38.4.2 Flexible SWJ-DP pin assignment 1686
38.4.3 Internal pull-up and pull-down on JTAG pins 1687
38.4.4 Using serial wire and releasing the unused debug pins as GPIOs . . 1688
38.5 STM32F4xx JTAG TAP connection 1688
38.6 ID codes and locking mechanism 1690
38.6.1 MCU device ID code 1690
38.6.2 Boundary scan TAP 1691
38.6.3 Cortex®-M4 with FPU TAP 1691
38.6.4 Cortex®-M4 with FPU JEDEC-106 ID code 1691
38.7 JTAG debug port 1691
38.8 SW debug port 1693
38.8.1 SW protocol introduction 1693
38.8.2 SW protocol sequence 1693

38.8.3 SW-DP state machine (reset, idle states, ID code) . . . 1694
38.8.4 DP and AP read/write accesses . . . 1694
38.8.5 SW-DP registers . . . 1695
38.8.6 SW-AP registers . . . 1696
38.9 AHB-AP (AHB access port) - valid for both JTAG-DP and SW-DP . . . 1696
38.10 Core debug . . . 1697
38.11 Capability of the debugger host to connect under system reset . . . 1698
38.12 FPB (Flash patch breakpoint) . . . 1698
38.13 DWT (data watchpoint trigger) . . . 1699
38.14 ITM (instrumentation trace macrocell) . . . 1699
38.14.1 General description . . . 1699
38.14.2 Time stamp packets, synchronization and overflow packets . . . 1699
38.15 ETM (Embedded trace macrocell) . . . 1701
38.15.1 ETM general description . . . 1701
38.15.2 ETM signal protocol and packet types . . . 1701
38.15.3 Main ETM registers . . . 1702
38.15.4 ETM configuration example . . . 1702
38.16 MCU debug component (DBGMCU) . . . 1702
38.16.1 Debug support for low-power modes . . . 1702
38.16.2 Debug support for timers, watchdog, bxCAN and I^2C . . . 1703
38.16.3 Debug MCU configuration register . . . 1703
38.16.4 Debug MCU APB1 freeze register (DBGMCU_APB1_FZ) . . . 1705
38.16.5 Debug MCU APB2 Freeze register (DBGMCU_APB2_FZ) . . . 1706
38.17 TPIU (trace port interface unit) . . . 1707
38.17.1 Introduction . . . 1707
38.17.2 TRACE pin assignment . . . 1708
38.17.3 TPUI formatter . . . 1709
38.17.4 TPUI frame synchronization packets . . . 1710
38.17.5 Transmission of the synchronization frame packet . . . 1710
38.17.6 Synchronous mode . . . 1710
38.17.7 Asynchronous mode . . . 1711
38.17.8 TRACECLKIN connection inside the STM32F4xx . . . 1711
38.17.9 TPIU registers . . . 1711
38.17.10 Example of configuration . . . 1712
38.18 DBG register map . . . 1713

39 Device electronic signature 1714

39.1 Unique device ID register (96 bits) 1714

39.2 Flash size 1715

40 Revision history 1716

List of tables

Table 1. STM32F4xx register boundary addresses . . . 64
Table 2. Boot modes . . . 69
Table 3. Memory mapping vs. Boot mode/physical remap in STM32F405xx/07xx and STM32F415xx/17xx . . . 71
Table 4. Memory mapping vs. Boot mode/physical remap in STM32F42xxx and STM32F43xxx . . . 71
Table 5. Flash module organization (STM32F40x and STM32F41x) . . . 75
Table 6. Flash module - 2 Mbyte dual bank organization (STM32F42xxx and STM32F43xxx) . . . 77
Table 7. 1 Mbyte Flash memory single bank vs dual bank organization (STM32F42xxx and STM32F43xxx) . . . 78
Table 8. 1 Mbyte single bank Flash memory organization (STM32F42xxx and STM32F43xxx) . . . 78
Table 9. 1 Mbyte dual bank Flash memory organization (STM32F42xxx and STM32F43xxx) . . . 79
Table 10. Number of wait states according to CPU clock (HCLK) frequency (STM32F405xx/07xx and STM32F415xx/17xx) . . . 80
Table 11. Number of wait states according to CPU clock (HCLK) frequency (STM32F42xxx and STM32F43xxx) . . . 81
Table 12. Program/erase parallelism . . . 85
Table 13. Flash interrupt request . . . 88
Table 14. Option byte organization . . . 88
Table 15. Description of the option bytes (STM32F405xx/07xx and STM32F415xx/17xx) . . . 89
Table 16. Description of the option bytes (STM32F42xxx and STM32F43xxx) . . . 90
Table 17. Access versus read protection level . . . 94
Table 18. OTP area organization . . . 97
Table 19. Flash register map and reset values (STM32F405xx/07xx and STM32F415xx/17xx) . . . 111
Table 20. Flash register map and reset values (STM32F42xxx and STM32F43xxx) . . . 111
Table 21. CRC calculation unit register map and reset values . . . 115
Table 22. Voltage regulator configuration mode versus device operating mode . . . 122
Table 23. Low-power mode summary . . . 128
Table 24. Sleep-now entry and exit . . . 129
Table 25. Sleep-on-exit entry and exit . . . 130
Table 26. Stop operating modes (STM32F405xx/07xx and STM32F415xx/17xx) . . . 131
Table 27. Stop mode entry and exit (for STM32F405xx/07xx and STM32F415xx/17xx) . . . 132
Table 28. Stop operating modes (STM32F42xxx and STM32F43xxx) . . . 134
Table 29. Stop mode entry and exit (STM32F42xxx and STM32F43xxx) . . . 136
Table 30. Standby mode entry and exit . . . 137
Table 31. PWR - register map and reset values for STM32F405xx/07xx and STM32F415xx/17xx . . . 149
Table 32. PWR - register map and reset values for STM32F42xxx and STM32F43xxx . . . 149
Table 33. RCC register map and reset values for STM32F42xxx and STM32F43xxx . . . 210
Table 34. RCC register map and reset values . . . 265
Table 35. Port bit configuration table . . . 268
Table 36. Flexible SWJ-DP pin assignment . . . 271
Table 37. RTC_AF1 pin . . . 279
Table 38. RTC_AF2 pin . . . 280
Table 39. GPIO register map and reset values . . . 287

Table 40. SYSCFG register map and reset values (STM32F405xx/07xx and STM32F415xx/17xx) 294
Table 41. SYSCFG register map and reset values (STM32F42xxx and STM32F43xxx) 301
Table 42. DMA1 request mapping 307
Table 43. DMA2 request mapping 308
Table 44. Source and destination address 309
Table 45. Source and destination address registers in Double buffer mode (DBM=1) 314
Table 46. Packing/unpacking & endian behavior (bit PINC = MINC = 1) 315
Table 47. Restriction on NDT versus PSIZE and MSIZE 316
Table 48. FIFO threshold configurations 318
Table 49. Possible DMA configurations 322
Table 50. DMA interrupt requests 324
Table 51. DMA register map and reset values 335
Table 52. Supported color mode in input 342
Table 53. Data order in memory 343
Table 54. Alpha mode configuration 344
Table 55. Supported CLUT color mode 345
Table 56. CLUT data order in memory 345
Table 57. Supported color mode in output 346
Table 58. Data order in memory 346
Table 59. DMA2D interrupt requests 351
Table 60. DMA2D register map and reset values 369
Table 61. Vector table for STM32F405xx/07xx and STM32F415xx/17xx 372
Table 62. Vector table for STM32F42xxx and STM32F43xxx 375
Table 63. External interrupt/event controller register map and reset values 387
Table 64. External interrupt/event controller register map and reset values 387
Table 65. ADC pins 390
Table 66. Analog watchdog channel selection 393
Table 67. Configuring the trigger polarity 397
Table 68. External trigger for regular channels 398
Table 69. External trigger for injected channels 399
Table 70. ADC interrupts 414
Table 71. ADC global register map 430
Table 72. ADC register map and reset values for each ADC 431
Table 73. ADC register map and reset values (common ADC registers) 432
Table 74. DAC pins 434
Table 75. External triggers 437
Table 76. DAC register map 453
Table 77. DCMI pins 455
Table 78. DCMI signals 457
Table 79. Positioning of captured data bytes in 32-bit words (8-bit width) 458
Table 80. Positioning of captured data bytes in 32-bit words (10-bit width) 458
Table 81. Positioning of captured data bytes in 32-bit words (12-bit width) 459
Table 82. Positioning of captured data bytes in 32-bit words (14-bit width) 459
Table 83. Data storage in monochrome progressive video format 466
Table 84. Data storage in RGB progressive video format 466
Table 85. Data storage in YCbCr progressive video format 467
Table 86. DCMI interrupts 467
Table 87. DCMI register map and reset values 478
Table 88. LTDC registers versus clock domain 482
Table 89. LCD-TFT pins and signal interface 483
Table 90. Pixel Data mapping versus Color Format 487
Table 91. LTDC interrupt requests 491

Table 92. LTDC register map and reset values . . . 512
Table 93. Counting direction versus encoder signals . . . 553
Table 94. TIMx Internal trigger connection . . . 567
Table 95. Output control bits for complementary OCx and OCxN channels with break feature . . . 579
Table 96. TIM1 and TIM8 register map and reset values . . . 587
Table 97. Counting direction versus encoder signals . . . 616
Table 98. TIMx internal trigger connection . . . 632
Table 99. Output control bit for standard OCx channels . . . 641
Table 100. TIM2 to TIM5 register map and reset values . . . 648
Table 101. TIMx internal trigger connection . . . 674
Table 102. Output control bit for standard OCx channels . . . 682
Table 103. TIM9/12 register map and reset values . . . 684
Table 104. Output control bit for standard OCx channels . . . 691
Table 105. TIM10/11/13/14 register map and reset values . . . 694
Table 106. TIM6 and TIM7 register map and reset values . . . 707
Table 107. Min/max IWDG timeout period (in ms) at 32 kHz (LSI) . . . 709
Table 108. IWDG register map and reset values . . . 712
Table 109. Minimum and maximum timeout values at 30 MHz (f_{PCLK1}) . . . 716
Table 110. WWDG register map and reset values . . . 719
Table 111. Number of cycles required to process each 128-bit block (STM32F415/417xx) . . . 720
Table 112. Number of cycles required to process each 128-bit block (STM32F43xxx) . . . 720
Table 113. Data types . . . 740
Table 114. CRYP register map and reset values for STM32F415/417xx . . . 763
Table 115. CRYP register map and reset values for STM32F43xxx . . . 764
Table 116. RNG register map and reset map . . . 771
Table 117. HASH register map and reset values on STM32F415/417xx . . . 795
Table 118. HASH register map and reset values on STM32F43xxx . . . 796
Table 119. Effect of low-power modes on RTC . . . 814
Table 120. Interrupt control bits . . . 815
Table 121. RTC register map and reset values . . . 836
Table 122. Maximum DNF[3:0] value to be compliant with Thd:dat(max) . . . 852
Table 123. SMBus vs. I2C . . . 854
Table 124. I2C Interrupt requests . . . 858
Table 125. I2C register map and reset values . . . 872
Table 126. SPI interrupt requests . . . 898
Table 127. Audio frequency precision (for PLLM VCO = 1 MHz or 2 MHz) . . . 909
Table 128. I^2S interrupt requests . . . 915
Table 129. SPI register map and reset values . . . 925
Table 130. Example of possible audio frequency sampling range . . . 936
Table 131. Interrupt sources . . . 948
Table 132. SAI register map and reset values . . . 963
Table 133. Noise detection from sampled data . . . 977
Table 134. Error calculation for programmed baud rates at f_{PCLK} = 8 MHz or f_{PCLK} = 12 MHz, oversampling by 16 . . . 980
Table 135. Error calculation for programmed baud rates at f_{PCLK} = 8 MHz or f_{PCLK} =12 MHz, oversampling by 8 . . . 981
Table 136. Error calculation for programmed baud rates at f_{PCLK} = 16 MHz or f_{PCLK} = 24 MHz, oversampling by 16 . . . 981
Table 137. Error calculation for programmed baud rates at f_{PCLK} = 16 MHz or f_{PCLK} = 24 MHz,

oversampling by 8. . . . 982
Table 138. Error calculation for programmed baud rates at fPCLK = 8 MHz or fPCLK = 16 MHz, oversampling by 16. . . . 983
Table 139. Error calculation for programmed baud rates at fPCLK = 8 MHz or fPCLK = 16 MHz, oversampling by 8. . . . 983
Table 140. Error calculation for programmed baud rates at fPCLK = 30 MHz or fPCLK = 60 MHz, oversampling by 16. . . . 984
Table 141. Error calculation for programmed baud rates at fPCLK = 30 MHz or fPCLK = 60 MHz, oversampling by 8 985
Table 142. Error calculation for programmed baud rates at fPCLK = 42 MHz or fPCLK = 84 Hz, oversampling by 16. . . . 986
Table 143. Error calculation for programmed baud rates at fPCLK = 42 MHz or fPCLK = 84 MHz, oversampling by 8. . . . 987
Table 144. USART receiver's tolerance when DIV fraction is 0 988
Table 145. USART receiver tolerance when DIV_Fraction is different from 0 989
Table 146. Frame formats 991
Table 147. USART interrupt requests. . . . 1006
Table 148. USART mode configuration 1007
Table 149. USART register map and reset values 1018
Table 150. SDIO I/O definitions 1023
Table 151. Command format 1027
Table 152. Short response format 1028
Table 153. Long response format. . . . 1028
Table 154. Command path status flags 1028
Table 155. Data token format 1031
Table 156. Transmit FIFO status flags 1032
Table 157. Receive FIFO status flags 1033
Table 158. Card status 1043
Table 159. SD status 1046
Table 160. Speed class code field 1047
Table 161. Performance move field 1047
Table 162. AU_SIZE field 1048
Table 163. Maximum AU size. . . . 1048
Table 164. Erase size field 1048
Table 165. Erase timeout field 1049
Table 166. Erase offset field. . . . 1049
Table 167. Block-oriented write commands 1051
Table 168. Block-oriented write protection commands. . . . 1052
Table 169. Erase commands 1052
Table 170. I/O mode commands 1053
Table 171. Lock card 1053
Table 172. Application-specific commands 1053
Table 173. R1 response 1054
Table 174. R2 response 1055
Table 175. R3 response 1055
Table 176. R4 response 1055
Table 177. R4b response 1056
Table 178. R5 response 1056
Table 179. R6 response 1057
Table 180. Response type and SDIO_RESPx registers. . . . 1064
Table 181. SDIO register map 1074
Table 182. Transmit mailbox mapping 1091

Table 183. Receive mailbox mapping . . . 1091
Table 184. bxCAN register map and reset values . . . 1118
Table 185. Alternate function mapping . . . 1125
Table 186. Management frame format . . . 1127
Table 187. Clock range . . . 1129
Table 188. TX interface signal encoding . . . 1130
Table 189. RX interface signal encoding . . . 1131
Table 190. Frame statuses . . . 1146
Table 191. Destination address filtering . . . 1152
Table 192. Source address filtering . . . 1153
Table 193. Receive descriptor 0 - encoding for bits 7, 5 and 0 (normal descriptor format only, EDFE=0) . . . 1184
Table 194. Time stamp snapshot dependency on registers bits . . . 1217
Table 195. Ethernet register map and reset values . . . 1236
Table 196. OTG_FS input/output pins . . . 1243
Table 197. Compatibility of STM32 low power modes with the OTG . . . 1256
Table 198. Core global control and status registers (CSRs) . . . 1266
Table 199. Host-mode control and status registers (CSRs) . . . 1267
Table 200. Device-mode control and status registers . . . 1268
Table 201. Data FIFO (DFIFO) access register map . . . 1269
Table 202. Power and clock gating control and status registers . . . 1270
Table 203. TRDT values . . . 1276
Table 204. Minimum duration for soft disconnect . . . 1304
Table 205. OTG_FS register map and reset values . . . 1326
Table 206. OTG_HS input/output pins . . . 1384
Table 207. Compatibility of STM32 low power modes with the OTG . . . 1397
Table 208. Core global control and status registers (CSRs) . . . 1403
Table 209. Host-mode control and status registers (CSRs) . . . 1404
Table 210. Device-mode control and status registers . . . 1405
Table 211. Data FIFO (DFIFO) access register map . . . 1407
Table 212. Power and clock gating control and status registers . . . 1407
Table 213. TRDT values . . . 1414
Table 214. Minimum duration for soft disconnect . . . 1446
Table 215. OTG_HS register map and reset values . . . 1472
Table 216. NOR/PSRAM bank selection . . . 1548
Table 217. External memory address . . . 1548
Table 218. Memory mapping and timing registers . . . 1548
Table 219. NAND bank selections . . . 1549
Table 220. Programmable NOR/PSRAM access parameters . . . 1550
Table 221. Nonmultiplexed I/O NOR Flash . . . 1550
Table 222. Multiplexed I/O NOR Flash . . . 1551
Table 223. Nonmultiplexed I/Os PSRAM/SRAM . . . 1551
Table 224. Multiplexed I/O PSRAM . . . 1552
Table 225. NOR Flash/PSRAM controller: example of supported memories and transactions . . . 1552
Table 226. FSMC_BCRx bit fields . . . 1555
Table 227. FSMC_BTRx bit fields . . . 1556
Table 228. FSMC_BCRx bit fields . . . 1557
Table 229. FSMC_BTRx bit fields . . . 1558
Table 230. FSMC_BWTRx bit fields . . . 1558
Table 231. FSMC_BCRx bit fields . . . 1560
Table 232. FSMC_BTRx bit fields . . . 1561
Table 233. FSMC_BWTRx bit fields . . . 1561

Table 234. FSMC_BCRx bit fields . . . 1563
Table 235. FSMC_BTRx bit fields . . . 1563
Table 236. FSMC_BWTRx bit fields . . . 1564
Table 237. FSMC_BCRx bit fields . . . 1565
Table 238. FSMC_BTRx bit fields . . . 1566
Table 239. FSMC_BWTRx bit fields . . . 1566
Table 240. FSMC_BCRx bit fields . . . 1568
Table 241. FSMC_BTRx bit fields . . . 1568
Table 242. FSMC_BCRx bit fields . . . 1573
Table 243. FSMC_BTRx bit fields . . . 1574
Table 244. FSMC_BCRx bit fields . . . 1575
Table 245. FSMC_BTRx bit fields . . . 1576
Table 246. Programmable NAND/PC Card access parameters . . . 1585
Table 247. 8-bit NAND Flash . . . 1585
Table 248. 16-bit NAND Flash . . . 1586
Table 249. 16-bit PC Card . . . 1586
Table 250. Supported memories and transactions . . . 1587
Table 251. 16-bit PC-Card signals and access type . . . 1592
Table 252. ECC result relevant bits . . . 1599
Table 253. FSMC register map . . . 1600
Table 254. NOR/PSRAM bank selection . . . 1607
Table 255. NOR/PSRAM External memory address . . . 1608
Table 256. NAND/PC Card memory mapping and timing registers . . . 1608
Table 257. NAND bank selection . . . 1609
Table 258. SDRAM bank selection . . . 1609
Table 259. SDRAM address mapping . . . 1609
Table 260. SDRAM address mapping with 8-bit data bus width . . . 1610
Table 261. SDRAM address mapping with 16-bit data bus width . . . 1611
Table 262. SDRAM address mapping with 32-bit data bus width . . . 1611
Table 263. Programmable NOR/PSRAM access parameters . . . 1613
Table 264. Non-multiplexed I/O NOR Flash memory . . . 1614
Table 265. 16-bit multiplexed I/O NOR Flash memory . . . 1614
Table 266. Non-multiplexed I/Os PSRAM/SRAM . . . 1614
Table 267. 16-Bit multiplexed I/O PSRAM . . . 1615
Table 268. NOR Flash/PSRAM: Example of supported memories and transactions . . . 1616
Table 269. FMC_BCRx bit fields . . . 1619
Table 270. FMC_BTRx bit fields . . . 1619
Table 271. FMC_BCRx bit fields . . . 1621
Table 272. FMC_BTRx bit fields . . . 1622
Table 273. FMC_BWTRx bit fields . . . 1622
Table 274. FMC_BCRx bit fields . . . 1624
Table 275. FMC_BTRx bit fields . . . 1625
Table 276. FMC_BWTRx bit fields . . . 1625
Table 277. FMC_BCRx bit fields . . . 1627
Table 278. FMC_BTRx bit fields . . . 1627
Table 279. FMC_BWTRx bit fields . . . 1628
Table 280. FMC_BCRx bit fields . . . 1629
Table 281. FMC_BTRx bit fields . . . 1630
Table 282. FMC_BWTRx bit fields . . . 1630
Table 283. FMC_BCRx bit fields . . . 1632
Table 284. FMC_BTRx bit fields . . . 1632
Table 285. FMC_BCRx bit fields . . . 1637

Table 286. FMC_BTRx bit fields . . . 1637
Table 287. FMC_BCRx bit fields . . . 1638
Table 288. FMC_BTRx bit fields . . . 1639
Table 289. Programmable NAND Flash/PC Card access parameters . . . 1648
Table 290. 8-bit NAND Flash . . . 1648
Table 291. 16-bit NAND Flash . . . 1649
Table 292. 16-bit PC Card . . . 1649
Table 293. Supported memories and transactions . . . 1650
Table 294. 16-bit PC-Card signals and access type . . . 1655
Table 295. ECC result relevant bits . . . 1662
Table 296. SDRAM signals . . . 1663
Table 297. FMC register map . . . 1680
Table 298. SWJ debug port pins . . . 1686
Table 299. Flexible SWJ-DP pin assignment . . . 1686
Table 300. JTAG debug port data registers . . . 1691
Table 301. 32-bit debug port registers addressed through the shifted value A[3:2] . . . 1692
Table 302. Packet request (8-bits) . . . 1693
Table 303. ACK response (3 bits) . . . 1694
Table 304. DATA transfer (33 bits) . . . 1694
Table 305. SW-DP registers . . . 1695
Table 306. Cortex®-M4 with FPU AHB-AP registers . . . 1696
Table 307. Core debug registers . . . 1697
Table 308. Main ITM registers . . . 1700
Table 309. Main ETM registers . . . 1702
Table 310. Asynchronous TRACE pin assignment . . . 1708
Table 311. Synchronous TRACE pin assignment . . . 1708
Table 312. Flexible TRACE pin assignment . . . 1709
Table 313. Important TPIU registers . . . 1711
Table 314. DBG register map and reset values . . . 1713
Table 315. Document revision history . . . 1716

List of figures

Figure 1. System architecture for STM32F405xx/07xx and STM32F415xx/17xx devices 60
Figure 2. System architecture for STM32F42xxx and STM32F43xxx devices 62
Figure 3. Flash memory interface connection inside system architecture (STM32F405xx/07xx and STM32F415xx/17xx) 73
Figure 4. Flash memory interface connection inside system architecture (STM32F42xxx and STM32F43xxx) 74
Figure 5. Sequential 32-bit instruction execution 83
Figure 6. RDP levels 94
Figure 7. PCROP levels 96
Figure 8. CRC calculation unit block diagram 113
Figure 9. Power supply overview for STM32F405xx/07xx and STM32F415xx/17xx 116
Figure 10. Power supply overview for STM32F42xxx and STM32F43xxx 117
Figure 11. Backup domain 120
Figure 12. Power-on reset/power-down reset waveform 124
Figure 13. BOR thresholds 125
Figure 14. PVD thresholds 126
Figure 15. Simplified diagram of the reset circuit 151
Figure 16. Clock tree 152
Figure 17. HSE/ LSE clock sources 154
Figure 18. Frequency measurement with TIM5 in Input capture mode 159
Figure 19. Frequency measurement with TIM11 in Input capture mode 160
Figure 20. Simplified diagram of the reset circuit 214
Figure 21. Clock tree 216
Figure 22. HSE/ LSE clock sources 218
Figure 23. Frequency measurement with TIM5 in Input capture mode 223
Figure 24. Frequency measurement with TIM11 in Input capture mode 223
Figure 25. Basic structure of a five-volt tolerant I/O port bit 268
Figure 26. Selecting an alternate function on STM32F405xx/07xx and STM32F415xx/17xx 272
Figure 27. Selecting an alternate function on STM32F42xxx and STM32F43xxx 273
Figure 28. Input floating/pull up/pull down configurations 276
Figure 29. Output configuration 277
Figure 30. Alternate function configuration 277
Figure 31. High impedance-analog configuration 278
Figure 32. DMA block diagram 304
Figure 33. System implementation of the two DMA controllers (STM32F405xx/07xx and STM32F415xx/17xx) 305
Figure 34. System implementation of the two DMA controllers (STM32F42xxx and STM32F43xxx) 306
Figure 35. Channel selection 307
Figure 36. Peripheral-to-memory mode 310
Figure 37. Memory-to-peripheral mode 311
Figure 38. Memory-to-memory mode 312
Figure 39. FIFO structure 317
Figure 40. DMA2D block diagram 341
Figure 41. External interrupt/event controller block diagram 380
Figure 42. External interrupt/event GPIO mapping (STM32F405xx/07xx and STM32F415xx/17xx). 382
Figure 43. External interrupt/event GPIO mapping (STM32F42xxx and STM32F43xxx) 383
Figure 44. Single ADC block diagram 389
Figure 45. Timing diagram 392

Figure 46. Analog watchdog's guarded area . . . 393
Figure 47. Injected conversion latency . . . 394
Figure 48. Right alignment of 12-bit data . . . 396
Figure 49. Left alignment of 12-bit data . . . 396
Figure 50. Left alignment of 6-bit data . . . 396
Figure 51. Multi ADC block diagram(1) . . . 402
Figure 52. Injected simultaneous mode on 4 channels: dual ADC mode . . . 405
Figure 53. Injected simultaneous mode on 4 channels: triple ADC mode . . . 405
Figure 54. Regular simultaneous mode on 16 channels: dual ADC mode . . . 406
Figure 55. Regular simultaneous mode on 16 channels: triple ADC mode . . . 406
Figure 56. Interleaved mode on 1 channel in continuous conversion mode: dual ADC mode . . . 407
Figure 57. Interleaved mode on 1 channel in continuous conversion mode: triple ADC mode . . . 408
Figure 58. Alternate trigger: injected group of each ADC . . . 409
Figure 59. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode . . . 410
Figure 60. Alternate trigger: injected group of each ADC . . . 410
Figure 61. Alternate + regular simultaneous . . . 411
Figure 62. Case of trigger occurring during injected conversion . . . 412
Figure 63. Temperature sensor and VREFINT channel block diagram . . . 413
Figure 64. DAC channel block diagram . . . 434
Figure 65. Data registers in single DAC channel mode . . . 436
Figure 66. Data registers in dual DAC channel mode . . . 436
Figure 67. Timing diagram for conversion with trigger disabled TEN = 0 . . . 437
Figure 68. DAC LFSR register calculation algorithm . . . 439
Figure 69. DAC conversion (SW trigger enabled) with LFSR wave generation . . . 439
Figure 70. DAC triangle wave generation . . . 440
Figure 71. DAC conversion (SW trigger enabled) with triangle wave generation . . . 440
Figure 72. DCMI block diagram . . . 456
Figure 73. Top-level block diagram . . . 457
Figure 74. DCMI signal waveforms . . . 458
Figure 75. Timing diagram . . . 460
Figure 76. Frame capture waveforms in Snapshot mode . . . 462
Figure 77. Frame capture waveforms in continuous grab mode . . . 463
Figure 78. Coordinates and size of the window after cropping . . . 464
Figure 79. Data capture waveforms . . . 464
Figure 80. Pixel raster scan order . . . 465
Figure 81. LTDC block diagram . . . 481
Figure 82. LCD-TFT Synchronous timings . . . 484
Figure 83. Layer window programmable parameters: . . . 487
Figure 84. Blending two layers with background . . . 490
Figure 85. Interrupt events . . . 491
Figure 86. Advanced-control timer block diagram . . . 517
Figure 87. Counter timing diagram with prescaler division change from 1 to 2 . . . 519
Figure 88. Counter timing diagram with prescaler division change from 1 to 4 . . . 519
Figure 89. Counter timing diagram, internal clock divided by 1 . . . 520
Figure 90. Counter timing diagram, internal clock divided by 2 . . . 521
Figure 91. Counter timing diagram, internal clock divided by 4 . . . 521
Figure 92. Counter timing diagram, internal clock divided by N . . . 521
Figure 93. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) . . . 522
Figure 94. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) . . . 522
Figure 95. Counter timing diagram, internal clock divided by 1 . . . 524
Figure 96. Counter timing diagram, internal clock divided by 2 . . . 524
Figure 97. Counter timing diagram, internal clock divided by 4 . . . 525

Figure 98. Counter timing diagram, internal clock divided by N . . . 525
Figure 99. Counter timing diagram, update event when repetition counter is not used . . . 526
Figure 100. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 . . . 527
Figure 101. Counter timing diagram, internal clock divided by 2 . . . 527
Figure 102. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 . . . 528
Figure 103. Counter timing diagram, internal clock divided by N . . . 528
Figure 104. Counter timing diagram, update event with ARPE=1 (counter underflow) . . . 529
Figure 105. Counter timing diagram, Update event with ARPE=1 (counter overflow) . . . 529
Figure 106. Update rate examples depending on mode and TIMx_RCR register settings . . . 531
Figure 107. Control circuit in normal mode, internal clock divided by 1 . . . 532
Figure 108. TI2 external clock connection example . . . 533
Figure 109. Control circuit in external clock mode 1 . . . 534
Figure 110. External trigger input block . . . 534
Figure 111. Control circuit in external clock mode 2 . . . 535
Figure 112. Capture/compare channel (example: channel 1 input stage) . . . 536
Figure 113. Capture/compare channel 1 main circuit . . . 536
Figure 114. Output stage of capture/compare channel (channel 1 to 3) . . . 537
Figure 115. Output stage of capture/compare channel (channel 4) . . . 537
Figure 116. PWM input mode timing . . . 539
Figure 117. Output compare mode, toggle on OC1 . . . 541
Figure 118. Edge-aligned PWM waveforms (ARR=8) . . . 542
Figure 119. Center-aligned PWM waveforms (ARR=8) . . . 543
Figure 120. Complementary output with dead-time insertion . . . 545
Figure 121. Dead-time waveforms with delay greater than the negative pulse . . . 545
Figure 122. Dead-time waveforms with delay greater than the positive pulse . . . 545
Figure 123. Output behavior in response to a break . . . 548
Figure 124. Clearing TIMx OCxREF . . . 549
Figure 125. 6-step generation, COM example (OSSR=1) . . . 550
Figure 126. Example of one pulse mode . . . 551
Figure 127. Example of counter operation in encoder interface mode . . . 554
Figure 128. Example of encoder interface mode with TI1FP1 polarity inverted . . . 554
Figure 129. Example of Hall sensor interface . . . 556
Figure 130. Control circuit in reset mode . . . 557
Figure 131. Control circuit in gated mode . . . 558
Figure 132. Control circuit in trigger mode . . . 559
Figure 133. Control circuit in external clock mode 2 + trigger mode . . . 560
Figure 134. General-purpose timer block diagram . . . 590
Figure 135. Counter timing diagram with prescaler division change from 1 to 2 . . . 592
Figure 136. Counter timing diagram with prescaler division change from 1 to 4 . . . 592
Figure 137. Counter timing diagram, internal clock divided by 1 . . . 593
Figure 138. Counter timing diagram, internal clock divided by 2 . . . 593
Figure 139. Counter timing diagram, internal clock divided by 4 . . . 594
Figure 140. Counter timing diagram, internal clock divided by N . . . 594
Figure 141. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded) . . . 595
Figure 142. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded) . . . 595
Figure 143. Counter timing diagram, internal clock divided by 1 . . . 596
Figure 144. Counter timing diagram, internal clock divided by 2 . . . 597
Figure 145. Counter timing diagram, internal clock divided by 4 . . . 597
Figure 146. Counter timing diagram, internal clock divided by N . . . 597
Figure 147. Counter timing diagram, Update event . . . 598
Figure 148. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6 . . . 599
Figure 149. Counter timing diagram, internal clock divided by 2 . . . 599

Figure 150. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 600
Figure 151. Counter timing diagram, internal clock divided by N. 600
Figure 152. Counter timing diagram, Update event with ARPE=1 (counter underflow). 601
Figure 153. Counter timing diagram, Update event with ARPE=1 (counter overflow) 601
Figure 154. Control circuit in normal mode, internal clock divided by 1 602
Figure 155. TI2 external clock connection example. 603
Figure 156. Control circuit in external clock mode 1 604
Figure 157. External trigger input block 604
Figure 158. Control circuit in external clock mode 2 605
Figure 159. Capture/compare channel (example: channel 1 input stage) 605
Figure 160. Capture/compare channel 1 main circuit 606
Figure 161. Output stage of capture/compare channel (channel 1). 606
Figure 162. PWM input mode timing 608
Figure 163. Output compare mode, toggle on OC1. 610
Figure 164. Edge-aligned PWM waveforms (ARR=8) 611
Figure 165. Center-aligned PWM waveforms (ARR=8). 612
Figure 166. Example of one-pulse mode 613
Figure 167. Clearing TIMx OCxREF 615
Figure 168. Example of counter operation in encoder interface mode 617
Figure 169. Example of encoder interface mode with TI1FP1 polarity inverted 617
Figure 170. Control circuit in reset mode 618
Figure 171. Control circuit in gated mode 619
Figure 172. Control circuit in trigger mode. 620
Figure 173. Control circuit in external clock mode 2 + trigger mode 621
Figure 174. Master/Slave timer example 621
Figure 175. Gating timer 2 with OC1REF of timer 1 622
Figure 176. Gating timer 2 with Enable of timer 1 623
Figure 177. Triggering timer 2 with update of timer 1 624
Figure 178. Triggering timer 2 with Enable of timer 1 625
Figure 179. Triggering timer 1 and 2 with timer 1 TI1 input 626
Figure 180. General-purpose timer block diagram (TIM9 and TIM12) 651
Figure 181. General-purpose timer block diagram (TIM10/11/13/14) 652
Figure 182. Counter timing diagram with prescaler division change from 1 to 2 654
Figure 183. Counter timing diagram with prescaler division change from 1 to 4 654
Figure 184. Counter timing diagram, internal clock divided by 1 655
Figure 185. Counter timing diagram, internal clock divided by 2 656
Figure 186. Counter timing diagram, internal clock divided by 4 656
Figure 187. Counter timing diagram, internal clock divided by N. 656
Figure 188. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) 657
Figure 189. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) 657
Figure 190. Control circuit in normal mode, internal clock divided by 1 658
Figure 191. TI2 external clock connection example. 659
Figure 192. Control circuit in external clock mode 1 659
Figure 193. Capture/compare channel (example: channel 1 input stage) 660
Figure 194. Capture/compare channel 1 main circuit 661
Figure 195. Output stage of capture/compare channel (channel 1). 661
Figure 196. PWM input mode timing 663
Figure 197. Output compare mode, toggle on OC1. 665
Figure 198. Edge-aligned PWM waveforms (ARR=8) 666
Figure 199. Example of one pulse mode. 667
Figure 200. Control circuit in reset mode 669
Figure 201. Control circuit in gated mode 670

Figure 202. Control circuit in trigger mode . . . 670
Figure 203. Basic timer block diagram . . . 696
Figure 204. Counter timing diagram with prescaler division change from 1 to 2 . . . 698
Figure 205. Counter timing diagram with prescaler division change from 1 to 4 . . . 698
Figure 206. Counter timing diagram, internal clock divided by 1 . . . 699
Figure 207. Counter timing diagram, internal clock divided by 2 . . . 700
Figure 208. Counter timing diagram, internal clock divided by 4 . . . 700
Figure 209. Counter timing diagram, internal clock divided by N . . . 700
Figure 210. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) . . . 701
Figure 211. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) . . . 701
Figure 212. Control circuit in normal mode, internal clock divided by 1 . . . 702
Figure 213. Independent watchdog block diagram . . . 709
Figure 214. Watchdog block diagram . . . 714
Figure 215. Window watchdog timing diagram . . . 715
Figure 216. Block diagram (STM32F415/417xx) . . . 722
Figure 217. Block diagram (STM32F43xxx) . . . 723
Figure 218. DES/TDES-ECB mode encryption . . . 725
Figure 219. DES/TDES-ECB mode decryption . . . 725
Figure 220. DES/TDES-CBC mode encryption . . . 727
Figure 221. DES/TDES-CBC mode decryption . . . 728
Figure 222. AES-ECB mode encryption . . . 729
Figure 223. AES-ECB mode decryption . . . 730
Figure 224. AES-CBC mode encryption . . . 731
Figure 225. AES-CBC mode decryption . . . 732
Figure 226. AES-CTR mode encryption . . . 733
Figure 227. AES-CTR mode decryption . . . 734
Figure 228. Initial counter block structure for the Counter mode . . . 734
Figure 229. 64-bit block construction according to DATATYPE . . . 741
Figure 230. Initialization vectors use in the TDES-CBC encryption . . . 743
Figure 231. CRYP interrupt mapping diagram . . . 748
Figure 232. Block diagram . . . 767
Figure 233. Block diagram for STM32F415/417xx . . . 773
Figure 234. Block diagram for STM32F43xxx . . . 774
Figure 235. Bit, byte and half-word swapping . . . 776
Figure 236. HASH interrupt mapping diagram . . . 782
Figure 237. RTC block diagram . . . 800
Figure 238. I2C bus protocol . . . 841
Figure 239. I2C block diagram for STM32F40x/41x . . . 842
Figure 240. I2C block diagram for STM32F42x/43x . . . 843
Figure 241. Transfer sequence diagram for slave transmitter . . . 845
Figure 242. Transfer sequence diagram for slave receiver . . . 846
Figure 243. Transfer sequence diagram for master transmitter . . . 849
Figure 244. Transfer sequence diagram for master receiver . . . 850
Figure 245. I2C interrupt mapping diagram . . . 859
Figure 246. SPI block diagram . . . 876
Figure 247. Single master/ single slave application . . . 877
Figure 248. Data clock timing diagram . . . 879
Figure 249. TI mode - Slave mode, single transfer . . . 881
Figure 250. TI mode - Slave mode, continuous transfer . . . 882
Figure 251. TI mode - master mode, single transfer . . . 883

Figure 252. TI mode - master mode, continuous transfer . . . 884
Figure 253. TXE/RXNE/BSY behavior in Master / full-duplex mode (BIDIMODE=0 and RXONLY=0) in case of continuous transfers . . . 887
Figure 254. TXE/RXNE/BSY behavior in Slave / full-duplex mode (BIDIMODE=0, RXONLY=0) in case of continuous transfers . . . 888
Figure 255. TXE/BSY behavior in Master transmit-only mode (BIDIMODE=0 and RXONLY=0) in case of continuous transfers . . . 889
Figure 256. TXE/BSY in Slave transmit-only mode (BIDIMODE=0 and RXONLY=0) in case of continuous transfers . . . 889
Figure 257. RXNE behavior in receive-only mode (BIDIRMODE=0 and RXONLY=1) in case of continuous transfers . . . 890
Figure 258. TXE/BSY behavior when transmitting (BIDIRMODE=0 and RXONLY=0) in case of discontinuous transfers . . . 891
Figure 259. Transmission using DMA . . . 896
Figure 260. Reception using DMA . . . 896
Figure 261. TI mode frame format error detection . . . 898
Figure 262. I^2S block diagram . . . 899
Figure 263. I2S full duplex block diagram . . . 900
Figure 264. I^2S Philips protocol waveforms (16/32-bit full accuracy, CPOL = 0) . . . 902
Figure 265. I^2S Philips standard waveforms (24-bit frame with CPOL = 0) . . . 902
Figure 266. Transmitting 0x8EAA33 . . . 902
Figure 267. Receiving 0x8EAA33 . . . 903
Figure 268. I^2S Philips standard (16-bit extended to 32-bit packet frame with CPOL = 0) . . . 903
Figure 269. Example . . . 903
Figure 270. MSB justified 16-bit or 32-bit full-accuracy length with CPOL = 0 . . . 904
Figure 271. MSB justified 24-bit frame length with CPOL = 0 . . . 904
Figure 272. MSB justified 16-bit extended to 32-bit packet frame with CPOL = 0 . . . 904
Figure 273. LSB justified 16-bit or 32-bit full-accuracy with CPOL = 0 . . . 905
Figure 274. LSB justified 24-bit frame length with CPOL = 0 . . . 905
Figure 275. Operations required to transmit 0x3478AE . . . 905
Figure 276. Operations required to receive 0x3478AE . . . 906
Figure 277. LSB justified 16-bit extended to 32-bit packet frame with CPOL = 0 . . . 906
Figure 278. Example of LSB justified 16-bit extended to 32-bit packet frame . . . 906
Figure 279. PCM standard waveforms (16-bit) . . . 907
Figure 280. PCM standard waveforms (16-bit extended to 32-bit packet frame) . . . 907
Figure 281. Audio sampling frequency definition . . . 908
Figure 282. I^2S clock generator architecture . . . 908
Figure 283. Functional block diagram . . . 928
Figure 284. Audio frame . . . 930
Figure 285. FS role is start of frame + channel side identification (FSDEF = TRIS = 1) . . . 933
Figure 286. FS role is start of frame (FSDEF = 0) . . . 933
Figure 287. Slot size configuration with FBOFF = 0 in SAI_xSLOTR . . . 934
Figure 288. First bit offset . . . 934
Figure 289. Audio block clock generator overview . . . 935
Figure 290. AC'97 audio frame . . . 939
Figure 291. Data companding hardware in an audio block in the SAI . . . 941
Figure 292. Tristate strategy on SD output line on an inactive slot . . . 943
Figure 293. Tristate on output data line in a protocol like I2S . . . 944
Figure 294. Overrun detection error . . . 945
Figure 295. FIFO underrun event . . . 946
Figure 296. USART block diagram . . . 968
Figure 297. Word length programming . . . 969

Figure 298. Configurable stop bits . . . 971
Figure 299. TC/TXE behavior when transmitting . . . 972
Figure 300. Start bit detection when oversampling by 16 or 8 . . . 973
Figure 301. Data sampling when oversampling by 16 . . . 976
Figure 302. Data sampling when oversampling by 8 . . . 977
Figure 303. Mute mode using Idle line detection . . . 990
Figure 304. Mute mode using address mark detection . . . 990
Figure 305. Break detection in LIN mode (11-bit break length - LBDL bit is set) . . . 993
Figure 306. Break detection in LIN mode vs. Framing error detection . . . 994
Figure 307. USART example of synchronous transmission . . . 995
Figure 308. USART data clock timing diagram (M=0) . . . 995
Figure 309. USART data clock timing diagram (M=1) . . . 996
Figure 310. RX data setup/hold time . . . 996
Figure 311. ISO 7816-3 asynchronous protocol . . . 997
Figure 312. Parity error detection using the 1.5 stop bits . . . 998
Figure 313. IrDA SIR ENDEC- block diagram . . . 1000
Figure 314. IrDA data modulation (3/16) -Normal mode . . . 1000
Figure 315. Transmission using DMA . . . 1002
Figure 316. Reception using DMA . . . 1003
Figure 317. Hardware flow control between 2 USARTs . . . 1003
Figure 318. RTS flow control . . . 1004
Figure 319. CTS flow control . . . 1005
Figure 320. USART interrupt mapping diagram . . . 1006
Figure 321. SDIO “no response” and “no data” operations . . . 1020
Figure 322. SDIO (multiple) block read operation . . . 1020
Figure 323. SDIO (multiple) block write operation . . . 1021
Figure 324. SDIO sequential read operation . . . 1021
Figure 325. SDIO sequential write operation . . . 1021
Figure 326. SDIO block diagram . . . 1022
Figure 327. SDIO adapter . . . 1023
Figure 328. Control unit . . . 1024
Figure 329. SDIO adapter command path . . . 1025
Figure 330. Command path state machine (CPSM) . . . 1026
Figure 331. SDIO command transfer . . . 1027
Figure 332. Data path . . . 1029
Figure 333. Data path state machine (DPSM) . . . 1030
Figure 334. CAN network topology . . . 1077
Figure 335. Dual CAN block diagram . . . 1079
Figure 336. bxCAN operating modes . . . 1081
Figure 337. bxCAN in silent mode . . . 1082
Figure 338. bxCAN in loop back mode . . . 1082
Figure 339. bxCAN in combined mode . . . 1083
Figure 340. Transmit mailbox states . . . 1085
Figure 341. Receive FIFO states . . . 1086
Figure 342. Filter bank scale configuration - register organization . . . 1088
Figure 343. Example of filter numbering . . . 1089
Figure 344. Filtering mechanism - Example . . . 1090
Figure 345. CAN error state diagram . . . 1092
Figure 346. Bit timing . . . 1094
Figure 347. CAN frames . . . 1095
Figure 348. Event flags and interrupt generation . . . 1096
Figure 349. RX and TX mailboxes . . . 1107

Figure 350. ETH block diagram . . . 1126
Figure 351. SMI interface signals . . . 1127
Figure 352. MDIO timing and frame structure - Write cycle . . . 1128
Figure 353. MDIO timing and frame structure - Read cycle . . . 1129
Figure 354. Media independent interface signals . . . 1129
Figure 355. MII clock sources . . . 1131
Figure 356. Reduced media-independent interface signals . . . 1132
Figure 357. RMII clock sources . . . 1132
Figure 358. Clock scheme . . . 1133
Figure 359. Address field format . . . 1135
Figure 360. MAC frame format . . . 1136
Figure 361. Tagged MAC frame format . . . 1137
Figure 362. Transmission bit order . . . 1143
Figure 363. Transmission with no collision . . . 1143
Figure 364. Transmission with collision . . . 1144
Figure 365. Frame transmission in MMI and RMII modes . . . 1144
Figure 366. Receive bit order . . . 1148
Figure 367. Reception with no error . . . 1149
Figure 368. Reception with errors . . . 1149
Figure 369. Reception with false carrier indication . . . 1149
Figure 370. MAC core interrupt masking scheme . . . 1150
Figure 371. Wakeup frame filter register . . . 1154
Figure 372. Networked time synchronization . . . 1158
Figure 373. System time update using the Fine correction method . . . 1160
Figure 374. PTP trigger output to TIM2 ITR1 connection . . . 1162
Figure 375. PPS output . . . 1163
Figure 376. Descriptor ring and chain structure . . . 1164
Figure 377. TxDMA operation in Default mode . . . 1168
Figure 378. TxDMA operation in OSF mode . . . 1170
Figure 379. Normal transmit descriptor . . . 1171
Figure 380. Enhanced transmit descriptor . . . 1177
Figure 381. Receive DMA operation . . . 1179
Figure 382. Normal Rx DMA descriptor structure . . . 1181
Figure 383. Enhanced receive descriptor field format with IEEE1588 time stamp enabled . . . 1187
Figure 384. Interrupt scheme . . . 1190
Figure 385. Ethernet MAC remote wakeup frame filter register (ETH_MACRWUFFR) . . . 1200
Figure 386. OTG full-speed block diagram . . . 1243
Figure 387. OTG A-B device connection . . . 1245
Figure 388. USB peripheral-only connection . . . 1247
Figure 389. USB host-only connection . . . 1251
Figure 390. SOF connectivity . . . 1255
Figure 391. Updating OTG_FS_HFIR dynamically . . . 1258
Figure 392. Device-mode FIFO address mapping and AHB FIFO access mapping . . . 1259
Figure 393. Host-mode FIFO address mapping and AHB FIFO access mapping . . . 1260
Figure 394. Interrupt hierarchy . . . 1264
Figure 395. CSR memory map . . . 1266
Figure 396. Transmit FIFO write task . . . 1338
Figure 397. Receive FIFO read task . . . 1339
Figure 398. Normal bulk/control OUT/SETUP and bulk/control IN transactions . . . 1340
Figure 399. Bulk/control IN transactions . . . 1343
Figure 400. Normal interrupt OUT/IN transactions . . . 1345
Figure 401. Normal isochronous OUT/IN transactions . . . 1350

Figure 402. Receive FIFO packet read . . . 1356
Figure 403. Processing a SETUP packet . . . 1358
Figure 404. Bulk OUT transaction . . . 1365
Figure 405. TRDT max timing case . . . 1374
Figure 406. A-device SRP . . . 1375
Figure 407. B-device SRP . . . 1376
Figure 408. A-device HNP . . . 1377
Figure 409. B-device HNP . . . 1379
Figure 410. USB OTG interface block diagram . . . 1384
Figure 411. USB host-only connection . . . 1391
Figure 412. SOF trigger output to TIM2 ITR1 connection . . . 1396
Figure 413. Updating OTG_HS_HFIR dynamically . . . 1398
Figure 414. Interrupt hierarchy . . . 1401
Figure 415. CSR memory map . . . 1403
Figure 416. Transmit FIFO write task . . . 1492
Figure 417. Receive FIFO read task . . . 1493
Figure 418. Normal bulk/control OUT/SETUP and bulk/control IN transactions - DMA mode . . . 1494
Figure 419. Normal bulk/control OUT/SETUP and bulk/control IN transactions - Slave mode . . . 1495
Figure 420. Bulk/control IN transactions - DMA mode . . . 1498
Figure 421. Bulk/control IN transactions - Slave mode . . . 1499
Figure 422. Normal interrupt OUT/IN transactions - DMA mode . . . 1501
Figure 423. Normal interrupt OUT/IN transactions - Slave mode . . . 1502
Figure 424. Normal isochronous OUT/IN transactions - DMA mode . . . 1507
Figure 425. Normal isochronous OUT/IN transactions - Slave mode . . . 1508
Figure 426. Receive FIFO packet read in slave mode . . . 1519
Figure 427. Processing a SETUP packet . . . 1521
Figure 428. Slave mode bulk OUT transaction . . . 1528
Figure 429. TRDT max timing case . . . 1537
Figure 430. A-device SRP . . . 1538
Figure 431. B-device SRP . . . 1539
Figure 432. A-device HNP . . . 1540
Figure 433. B-device HNP . . . 1542
Figure 434. FSMC block diagram . . . 1545
Figure 435. FSMC memory banks . . . 1547
Figure 436. Mode1 read accesses . . . 1554
Figure 437. Mode1 write accesses . . . 1555
Figure 438. ModeA read accesses . . . 1556
Figure 439. ModeA write accesses . . . 1557
Figure 440. Mode2 and mode B read accesses . . . 1559
Figure 441. Mode2 write accesses . . . 1559
Figure 442. Mode B write accesses . . . 1560
Figure 443. Mode C read accesses . . . 1562
Figure 444. Mode C write accesses . . . 1562
Figure 445. Mode D read accesses . . . 1564
Figure 446. Mode D write accesses . . . 1565
Figure 447. Multiplexed read accesses . . . 1567
Figure 448. Multiplexed write accesses . . . 1567
Figure 449. Asynchronous wait during a read access . . . 1570
Figure 450. Asynchronous wait during a write access . . . 1570
Figure 451. Wait configurations . . . 1572

Figure 452. Synchronous multiplexed read mode - NOR, PSRAM (CRAM) 1573
Figure 453. Synchronous multiplexed write mode - PSRAM (CRAM) 1575
Figure 454. NAND/PC Card controller timing for common memory access 1588
Figure 455. Access to non ‘CE don’t care’ NAND-Flash 1589
Figure 456. FMC block diagram 1604
Figure 457. FMC memory banks 1607
Figure 458. Mode1 read access waveforms 1618
Figure 459. Mode1 write access waveforms 1618
Figure 460. ModeA read access waveforms 1620
Figure 461. ModeA write access waveforms 1621
Figure 462. Mode2 and mode B read access waveforms 1623
Figure 463. Mode2 write access waveforms 1623
Figure 464. ModeB write access waveforms 1624
Figure 465. ModeC read access waveforms 1626
Figure 466. ModeC write access waveforms 1626
Figure 467. ModeD read access waveforms 1628
Figure 468. ModeD write access waveforms 1629
Figure 469. Muxed read access waveforms 1631
Figure 470. Muxed write access waveforms 1631
Figure 471. Asynchronous wait during a read access waveforms 1633
Figure 472. Asynchronous wait during a write access waveforms 1634
Figure 473. Wait configuration waveforms 1636
Figure 474. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM) 1636
Figure 475. Synchronous multiplexed write mode waveforms - PSRAM (CRAM) 1638
Figure 476. NAND Flash/PC Card controller waveforms for common memory access 1651
Figure 477. Access to non ‘CE don’t care’ NAND-Flash 1652
Figure 478. Burst write SDRAM access waveforms 1665
Figure 479. Burst read SDRAM access 1666
Figure 480. Logic diagram of Read access with RBURST bit set (CAS=2, RPIPE=0) 1667
Figure 481. Read access crossing row boundary 1669
Figure 482. Write access crossing row boundary 1669
Figure 483. Self-refresh mode 1672
Figure 484. Power-down mode 1673
Figure 485. Block diagram of STM32 MCU and Cortex®-M4 with FPU-level debug support 1683
Figure 486. SWJ debug port 1685
Figure 487. JTAG TAP connections 1689
Figure 488. TPIU block diagram 1707

1 Documentation conventions

The STM32F401xx devices have an Arm®(a) Cortex®-M4 with FPU.

1.1 List of abbreviations for registers

The following abbreviations are used in register descriptions:

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

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

1.2 Glossary

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

- The CPU core integrates two debug ports:
  - JTAG debug port (JTAG-DP) provides a 5-pin standard interface based on the Joint Test Action Group (JTAG) protocol.
  - SWD debug port (SWD-DP) provides a 2-pin (clock and data) interface based on the Serial Wire Debug (SWD) protocol.

    For both the JTAG and SWD protocols, please refer to the Cortex®-M4 with FPU Technical Reference Manual
- Word: data/instruction of 32-bit length.
- Half word: data/instruction of 16-bit length.
- Byte: data of 8-bit length.
- Double word: data of 64-bit length.
- IAP (in-application programming): IAP is the ability to reprogram 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.
- I-Code: this bus connects the Instruction bus of the CPU core to the Flash instruction interface. Prefetch is performed on this bus.
- D-Code: this bus connects the D-Code bus (literal load and debug access) of the CPU to the Flash data interface.
- Option bytes: product configuration bits stored in the Flash memory.
- OBL: option byte loader.
- AHB: advanced high-performance bus.
- CPU: refers to the Cortex®-M4 with FPU core.

1.3 Peripheral availability

For peripheral availability and number across all STM32F405xx/07xx and STM32F415xx/17xx sales types, please refer to the STM32F405xx/07xx and STM32F415xx/17xx datasheets.

For peripheral availability and number across all STM32F42xxx and STM32F43xxx sales types, please refer to the STM32F42xxx and STM32F43xxx datasheets.

2 Memory and bus architecture

2.1 System architecture

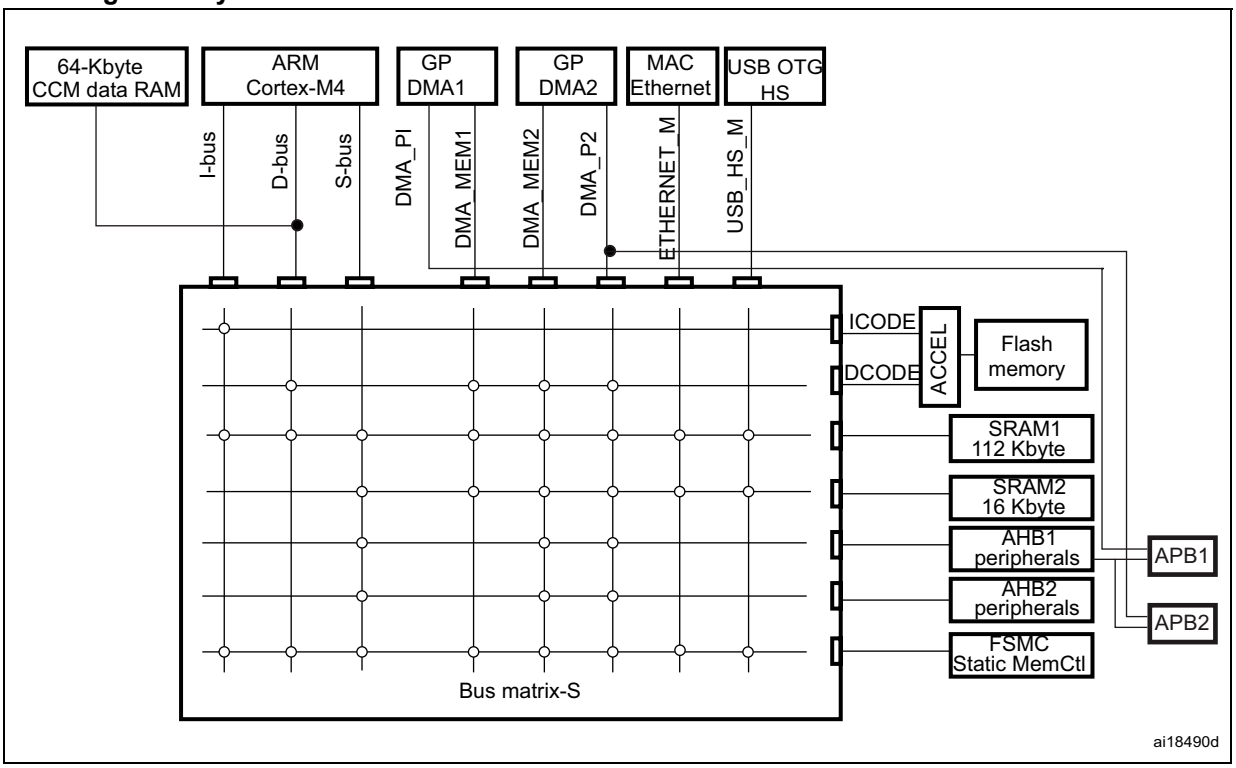
In STM32F405xx/07xx and STM32F415xx/17xx, the main system consists of 32-bit multilayer AHB bus matrix that interconnects:

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

- Eight masters:
  - Cortex®-M4 with FPU core I-bus, D-bus and S-bus
  - DMA1 memory bus
  - DMA2 memory bus
  - DMA2 peripheral bus
  - Ethernet DMA bus
  - USB OTG HS DMA bus
- Seven slaves:
  - Internal Flash memory ICode bus
  - Internal Flash memory DCode bus
  - Main internal SRAM1 (112 KB)
  - Auxiliary internal SRAM2 (16 KB)
  - AHB1 peripherals including AHB to APB bridges and APB peripherals
  - AHB2 peripherals
  - FSMC

The bus matrix provides access from a master to a slave, enabling concurrent access and efficient operation even when several high-speed peripherals work simultaneously. The 64-Kbyte CCM (core coupled memory) data RAM is not part of the bus matrix and can be accessed only through the CPU. This architecture is shown in *Figure 1*.

Figure 1. System architecture for STM32F405xx/07xx and STM32F415xx/17xx devices

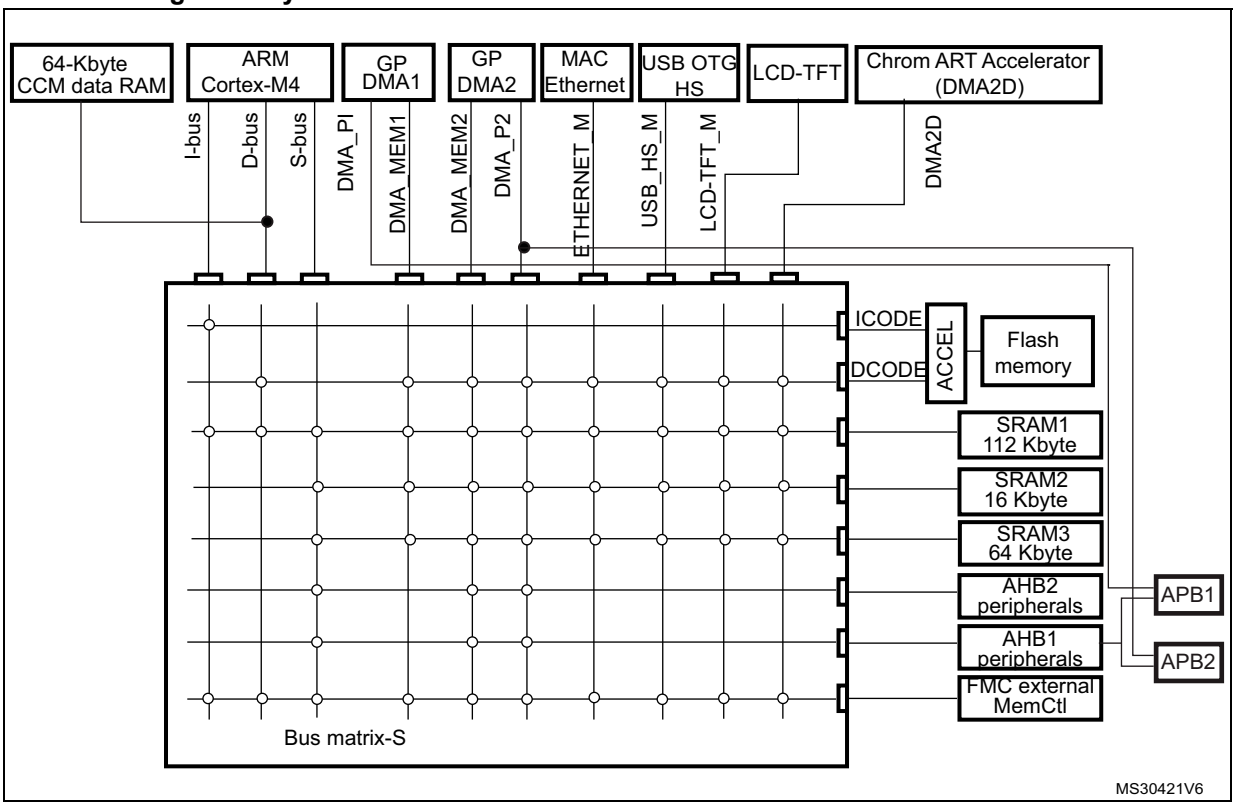


In the STM32F42xx and STM32F43xx devices, the main system consists of 32-bit multilayer AHB bus matrix that interconnects:

- Ten masters:
  - Cortex®-M4 with FPU core I-bus, D-bus and S-bus
  - DMA1 memory bus
  - DMA2 memory bus
  - DMA2 peripheral bus
  - Ethernet DMA bus
  - USB OTG HS DMA bus
  - LCD Controller DMA-bus
  - DMA2D (Chrom-Art Accelerator™) memory bus
- Eight slaves:
  - Internal Flash memory ICode bus
  - Internal Flash memory DCode bus
  - Main internal SRAM1 (112 KB)
  - Auxiliary internal SRAM2 (16 KB)
  - Auxiliary internal SRAM3 (64 KB)
  - AHB1peripherals including AHB to APB bridges and APB peripherals
  - AHB2 peripherals
  - FMC

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. The 64-Kbyte CCM (core coupled memory) data RAM is not part of the bus matrix and can be accessed only through the CPU. This architecture is shown in *Figure 2*.

Figure 2. System architecture for STM32F42xxx and STM32F43xxx devices



2.1.1 I-bus

This bus connects the Instruction bus of the Cortex®-M4 with FPU core to the BusMatrix. This bus is used by the core to fetch instructions. The target of this bus is a memory containing code (internal Flash memory/SRAM or external memories through the FSMC/FMC).

2.1.2 D-bus

This bus connects the databus of the Cortex®-M4 with FPU to the 64-Kbyte CCM data RAM to the BusMatrix. This bus is used by the core for literal load and debug access. The target of this bus is a memory containing code or data (internal Flash memory or external memories through the FSMC/FMC).

2.1.3 S-bus

This bus connects the system bus of the Cortex®-M4 with FPU core to a BusMatrix. This bus is used to access data located in a peripheral or in SRAM. Instructions may also be fetched on this bus (less efficient than ICode). The targets of this bus are the internal SRAM1, SRAM2 and SRAM3, the AHB1 peripherals including the APB peripherals, the AHB2 peripherals and the external memories through the FSMC/FMC.

2.1.4 DMA memory bus

This bus connects the DMA memory bus master interface to the BusMatrix. It is used by the DMA to perform transfer to/from memories. The targets of this bus are data memories: internal SRAMs (SRAM1, SRAM2 and SRAM3) and external memories through the FSMC/FMC.

2.1.5 DMA peripheral bus

This bus connects the DMA peripheral master bus interface to the BusMatrix. This bus is used by the DMA to access AHB peripherals or to perform memory-to-memory transfers. The targets of this bus are the AHB and APB peripherals plus data memories: internal SRAMs (SRAM1, SRAM2 and SRAM3) and external memories through the FSMC/FMC.

2.1.6 Ethernet DMA bus

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

2.1.7 USB OTG HS DMA bus

This bus connects the USB OTG HS DMA master interface to the BusMatrix. This bus is used by the USB OTG DMA to load/store data to a memory. The targets of this bus are data memories: internal SRAMs (SRAM1, SRAM2, SRAM3), internal Flash memory, and external memories through the FSMC/FMC.

2.1.8 LCD-TFT controller DMA bus

This bus connects the LCD controller DMA master interface to the BusMatrix. It is used by the LCD-TFT DMA to load/store data to a memory. The targets of this bus are data memories: internal SRAMs (SRAM1, SRAM2, SRAM3), external memories through FMC, and internal Flash memory.

2.1.9 DMA2D bus

This bus connect the DMA2D master interface to the BusMatrix. This bus is used by the DMA2D graphic Accelerator to load/store data to a memory. The targets of this bus are data memories: internal SRAMs (SRAM1, SRAM2, SRAM3), external memories through FMC, and internal Flash memory.

2.1.10 BusMatrix

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

2.1.11 AHB/APB bridges (APB)

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

Refer to the device datasheets for more details on APB1 and APB2 maximum frequencies, and to *Table 1* for the address mapping of AHB and APB peripherals.

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

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

2.2 Memory organization

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

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

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

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

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

2.3 Memory map

See the datasheet corresponding to your device for a comprehensive diagram of the memory map. *Table 1* gives the boundary addresses of the peripherals available in all STM32F4xx devices.

Table 1. STM32F4xx register boundary addresses

Boundary address	Peripheral	Bus	Register map
0xA000 0000 - 0xA000 0FFF	FSMC control register (STM32F405xx/07xx and STM32F415xx/17xx)/ FMC control register (STM32F42xxx and STM32F43xxx)	AHB3	*Section 36.6.9: FSMC register map on page 1600* *Section 37.8: FMC register map on page 1680*

Table 1. STM32F4xx register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x5006 0800 - 0x5006 0BFF	RNG	AHB2	*Section 24.4.4: RNG register map on page 771*
0x5006 0400 - 0x5006 07FF	HASH		*Section 25.4.9: HASH register map on page 795*
0x5006 0000 - 0x5006 03FF	CRYP		*Section 23.6.13: CRYP register map on page 763*
0x5005 0000 - 0x5005 03FF	DCMI		*Section 15.8.12: DCMI register map on page 478*
0x5000 0000 - 0x5003 FFFF	USB OTG FS		*Section 34.16.6: OTG_FS register map on page 1326*
0x4004 0000 - 0x4007 FFFF	USB OTG HS	AHB1	*Section 35.12.6: OTG_HS register map on page 1472*
0x4002 B000 - 0x4002 BBFF	DMA2D		*Section 11.5: DMA2D registers on page 352*
0x4002 8000 - 0x4002 93FF	ETHERNET MAC		*Section 33.8.5: Ethernet register maps on page 1236*
0x4002 6400 - 0x4002 67FF	DMA2		*Section 10.5.11: DMA register map on page 335*
0x4002 6000 - 0x4002 63FF	DMA1		
0x4002 4000 - 0x4002 4FFF	BKPSRAM		
0x4002 3C00 - 0x4002 3FFF	Flash interface register		*Section 3.9: Flash interface registers*
0x4002 3800 - 0x4002 3BFF	RCC		*Section 7.3.24: RCC register map on page 265*
0x4002 3000 - 0x4002 33FF	CRC		*Section 4.4.4: CRC register map on page 115*
0x4002 2800 - 0x4002 2BFF	GPIOK		*Section 8.4.11: GPIO register map on page 287*
0x4002 2400 - 0x4002 27FF	GPIOJ		
0x4002 2000 - 0x4002 23FF	GPIOI		*Section 8.4.11: GPIO register map on page 287*
0x4002 1C00 - 0x4002 1FFF	GPIOH		
0x4002 1800 - 0x4002 1BFF	GPIOG		
0x4002 1400 - 0x4002 17FF	GPIOF		
0x4002 1000 - 0x4002 13FF	GPIOE		
0x4002 0C00 - 0x4002 0FFF	GPIOD		
0x4002 0800 - 0x4002 0BFF	GPIOC		
0x4002 0400 - 0x4002 07FF	GPIOB		
0x4002 0000 - 0x4002 03FF	GPIOA		
0x4001 6800 - 0x4001 6BFF	LCD-TFT	APB2	*Section 16.7.26: LTDC register map on page 512*
0x4001 5800 - 0x4001 5BFF	SAI1		*Section 29.17.9: SAI register map on page 963*
0x4001 5400 - 0x4001 57FF	SPI6	APB2	*Section 28.5.10: SPI register map on page 925*
0x4001 5000 - 0x4001 53FF	SPI5		

Table 1. STM32F4xx register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x4001 4800 - 0x4001 4BFF	TIM11	APB2	*Section 19.5.12: TIM10/11/13/14 register map on page 694*
0x4001 4400 - 0x4001 47FF	TIM10		
0x4001 4000 - 0x4001 43FF	TIM9		*Section 19.4.13: TIM9/12 register map on page 684*
0x4001 3C00 - 0x4001 3FFF	EXTI		*Section 12.3.7: EXTI register map on page 387*
0x4001 3800 - 0x4001 3BFF	SYSCFG		*Section 9.2.8: SYSCFG register maps for STM32F405xx/07xx and STM32F415xx/17xx on page 294* and *Section 9.3.8: SYSCFG register maps for STM32F42xxx and STM32F43xxx on page 301*
0x4001 3400 - 0x4001 37FF	SPI4	APB2	*Section 28.5.10: SPI register map on page 925*
0x4001 3000 - 0x4001 33FF	SPI1	APB2	*Section 28.5.10: SPI register map on page 925*
0x4001 2C00 - 0x4001 2FFF	SDIO		*Section 31.9.16: SDIO register map on page 1074*
0x4001 2000 - 0x4001 23FF	ADC1 - ADC2 - ADC3		*Section 13.13.18: ADC register map on page 430*
0x4001 1400 - 0x4001 17FF	USART6		*Section 30.6.8: USART register map on page 1018*
0x4001 1000 - 0x4001 13FF	USART1		
0x4001 0400 - 0x4001 07FF	TIM8		*Section 17.4.21: TIM1 and TIM8 register map on page 587*
0x4001 0000 - 0x4001 03FF	TIM1		
0x4000 7C00 - 0x4000 7FFF	UART8	APB1	*Section 30.6.8: USART register map on page 1018*
0x4000 7800 - 0x4000 7BFF	UART7		

Table 1. STM32F4xx register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x4000 7400 - 0x4000 77FF	DAC	APB1	*Section 14.5.15: DAC register map on page 453*
0x4000 7000 - 0x4000 73FF	PWR		*Section 5.6: PWR register map on page 149*
0x4000 6800 - 0x4000 6BFF	CAN2		*Section 32.9.5: bxCAN register map on page 1118*
0x4000 6400 - 0x4000 67FF	CAN1		
0x4000 5C00 - 0x4000 5FFF	I2C3		*Section 27.6.11: I2C register map on page 872*
0x4000 5800 - 0x4000 5BFF	I2C2		
0x4000 5400 - 0x4000 57FF	I2C1		
0x4000 5000 - 0x4000 53FF	UART5		*Section 30.6.8: USART register map on page 1018*
0x4000 4C00 - 0x4000 4FFF	UART4		
0x4000 4800 - 0x4000 4BFF	USART3		
0x4000 4400 - 0x4000 47FF	USART2		
0x4000 4000 - 0x4000 43FF	I2S3ext		*Section 28.5.10: SPI register map on page 925*
0x4000 3C00 - 0x4000 3FFF	SPI3 / I2S3		
0x4000 3800 - 0x4000 3BFF	SPI2 / I2S2		
0x4000 3400 - 0x4000 37FF	I2S2ext		
0x4000 3000 - 0x4000 33FF	IWDG		*Section 21.4.5: IWDG register map on page 712*
0x4000 2C00 - 0x4000 2FFF	WWDG		*Section 22.6.4: WWDG register map on page 719*
0x4000 2800 - 0x4000 2BFF	RTC & BKP Registers		*Section 26.6.21: RTC register map on page 836*
0x4000 2000 - 0x4000 23FF	TIM14		*Section 19.5.12: TIM10/11/13/14 register map on page 694*
0x4000 1C00 - 0x4000 1FFF	TIM13		
0x4000 1800 - 0x4000 1BFF	TIM12		*Section 19.4.13: TIM9/12 register map on page 684*
0x4000 1400 - 0x4000 17FF	TIM7		*Section 20.4.9: TIM6 and TIM7 register map on page 707*
0x4000 1000 - 0x4000 13FF	TIM6		
0x4000 0C00 - 0x4000 0FFF	TIM5		*Section 18.4.21: TIMx register map on page 648*
0x4000 0800 - 0x4000 0BFF	TIM4		
0x4000 0400 - 0x4000 07FF	TIM3		
0x4000 0000 - 0x4000 03FF	TIM2		

2.3.1 Embedded SRAM

The STM32F405xx/07xx and STM32F415xx/17xx feature 4 Kbytes of backup SRAM (see *Section 5.1.2: Battery backup domain*) plus 192 Kbytes of system SRAM.

The STM32F42xxx and STM32F43xxx feature 4 Kbytes of backup SRAM (see *Section 5.1.2: Battery backup domain*) plus 256 Kbytes of system SRAM.

The embedded SRAM can be accessed as bytes, half-words (16 bits) or full words (32 bits). Read and write operations are performed at CPU speed with 0 wait state. The embedded SRAM is divided into up to three blocks:

- SRAM1 and SRAM2 mapped at address 0x2000 0000 and accessible by all AHB masters.
- SRAM3 (available on STM32F42xxx and STM32F43xxx) mapped at address 0x2002 0000 and accessible by all AHB masters.
- CCM (core coupled memory) mapped at address 0x1000 0000 and accessible only by the CPU through the D-bus.

The AHB masters support concurrent SRAM accesses (from the Ethernet or the USB OTG HS): for instance, the Ethernet MAC can read/write from/to SRAM2 while the CPU is reading/writing from/to SRAM1 or SRAM3.

The CPU can access the SRAM1, SRAM2, and SRAM3 through the System Bus or through the I-Code/D-Code buses when boot from SRAM is selected or when physical remap is selected (*Section 9.2.1: SYSCFG memory remap register (SYSCFG_MEMRMP)* in the SYSCFG controller). To get the max performance on SRAM execution, physical remap should be selected (boot or software selection).

2.3.2 Flash memory overview

The Flash memory interface manages CPU AHB I-Code and D-Code accesses to the Flash memory. It implements the erase and program Flash memory operations and the read and write protection mechanisms. It accelerates code execution with a system of instruction prefetch and cache lines.

The Flash memory is organized as follows:

- A main memory block divided into sectors.
- System memory from which the device boots in System memory boot mode
- 512 OTP (one-time programmable) bytes for user data.
- Option bytes to configure read and write protection, BOR level, watchdog software/hardware and reset when the device is in Standby or Stop mode.

Refer to *Section 3: Embedded Flash memory interface* for more details.

2.3.3 Bit banding

The Cortex®-M4 with FPU 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 STM32F4xx 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®-M4 with FPU 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:

bit_word_addr = *bit_band_base* + (*byte_offset* x 32) + (*bit_number* × 4)

where:

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

Example

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

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

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

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

For more information on bit-banding, please refer to the *Cortex®-M4 with FPU programming manual* (see *Related documents on page 1*).

2.4 Boot configuration

Due to its fixed memory map, the code area starts from address 0x0000 0000 (accessed through the ICode/DCode buses) while the data area (SRAM) starts from address 0x2000 0000 (accessed through the system bus). The Cortex®-M4 with FPU CPU always fetches the reset vector on the ICode bus, which implies to have the boot space available only in the code area (typically, Flash memory). STM32F4xx microcontrollers implement a special mechanism to be able to boot from other memories (like the internal SRAM).

In the STM32F4xx, three different boot modes can be selected through the BOOT[1:0] pins as shown in *Table 2*.

Table 2. Boot modes

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

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

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

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

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

In STM32F42xxx and STM32F43xxx devices, 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.

Embedded bootloader

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

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

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

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

Physical remap in STM32F405xx/07xx and STM32F415xx/17xx

Once the boot pins are selected, the application software can modify the memory accessible in the code area (in this way the code can be executed through the ICode bus in place of the System bus). This modification is performed by programming the *Section 9.2.1: SYSCFG memory remap register (SYSCFG_MEMRMP)* in the SYSCFG controller.

The following memories can thus be remapped:

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

Table 3. Memory mapping vs. Boot mode/physical remap in STM32F405xx/07xx and STM32F415xx/17xx

Addresses	Boot/Remap in main Flash memory	Boot/Remap in embedded SRAM	Boot/Remap in System memory	Remap in FSMC
0x2001 C000 - 0x2001 FFFF	SRAM2 (16 KB)	SRAM2 (16 KB)	SRAM2 (16 KB)	SRAM2 (16 KB)
0x2000 0000 - 0x2001 BFFF	SRAM1 (112 KB)	SRAM1 (112 KB)	SRAM1 (112 KB)	SRAM1 (112 KB)
0x1FFF 0000 - 0x1FFF 77FF	System memory	System memory	System memory	System memory
0x0810 0000 - 0x0FFF FFFF	Reserved	Reserved	Reserved	Reserved
0x0800 0000 - 0x080F FFFF	Flash memory	Flash memory	Flash memory	Flash memory
0x0400 0000 - 0x07FF FFFF	Reserved	Reserved	Reserved	FSMC bank 1 NOR/PSRAM 2 (128 MB Aliased)
0x0000 0000 - 0x000F FFFF(1)(2)	Flash (1 MB) Aliased	SRAM1 (112 KB) Aliased	System memory (30 KB) Aliased	FSMC bank 1 NOR/PSRAM 1 (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. 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.

Physical remap in STM32F42xxx and STM32F43xxx

Once the boot pins are selected, the application software can modify the memory accessible in the code area (in this way the code can be executed through the ICode bus in place of the System bus). This modification is performed by programming the *Section 9.2.1: SYSCFG memory remap register (SYSCFG_MEMRMP)* in the SYSCFG controller.

The following memories can thus be remapped:

- Main Flash memory
- System memory
- Embedded SRAM1 (112 KB)
- FMC bank 1 (NOR/PSRAM 1 and 2)
- FMC SDRAM bank 1

Table 4. Memory mapping vs. Boot mode/physical remap in STM32F42xxx and STM32F43xxx

Addresses	Boot/Remap in main Flash memory	Boot/Remap in embedded SRAM	Boot/Remap in System memory	Remap in FMC
0x2002 0000 - 0x2002 FFFF	SRAM3 (64 KB)	SRAM3 (64 KB)	SRAM3 (64 KB)	SRAM3 (64 KB)
0x2001 C000 - 0x2001 FFFF	SRAM2 (16 KB)	SRAM2 (16 KB)	SRAM2 (16 KB)	SRAM2 (16 KB)
0x2000 0000 - 0x2001 BFFF	SRAM1 (112 KB)	SRAM1 (112 KB)	SRAM1 (112 KB)	SRAM1 (112 KB)
0x1FFF 0000 - 0x1FFF 77FF	System memory	System memory	System memory	System memory
0x0810 0000 - 0x0FFF FFFF	Reserved	Reserved	Reserved	Reserved
0x0800 0000 - 0x081F FFFF	Flash memory	Flash memory	Flash memory	Flash memory

Table 4. Memory mapping vs. Boot mode/physical remap in STM32F42xxx and STM32F43xxx (continued)

Addresses	Boot/Remap in main Flash memory	Boot/Remap in embedded SRAM	Boot/Remap in System memory	Remap in FMC
0x0400 0000 - 0x07FF FFFF	Reserved	Reserved	Reserved	FMC bank 1 NOR/PSRAM 2 (128 MB Aliased)
0x0000 0000 - 0x001F FFFF(1)(2)	Flash (2 MB) Aliased	SRAM1 (112 KB) Aliased	System memory (30 KB) Aliased	FMC bank 1 NOR/PSRAM 1 (128 MB Aliased) or FMC SDRAM bank 1 (128 MB Aliased)

1. When the FMC 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) or SDRAM bank 1 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.
2. Even when aliased in the boot memory space, the related memory is still accessible at its original memory space.

3 Embedded Flash memory interface

3.1 Introduction

The Flash memory interface manages CPU AHB I-Code and D-Code 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 Main features

- Flash memory read operations
- Flash memory program/erase operations
- Read / write protections
- Prefetch on I-Code
- 64 cache lines of 128 bits on I-Code
- 8 cache lines of 128 bits on D-Code

Figure 3 shows the Flash memory interface connection inside the system architecture.

Figure 3. Flash memory interface connection inside system architecture (STM32F405xx/07xx and STM32F415xx/17xx)

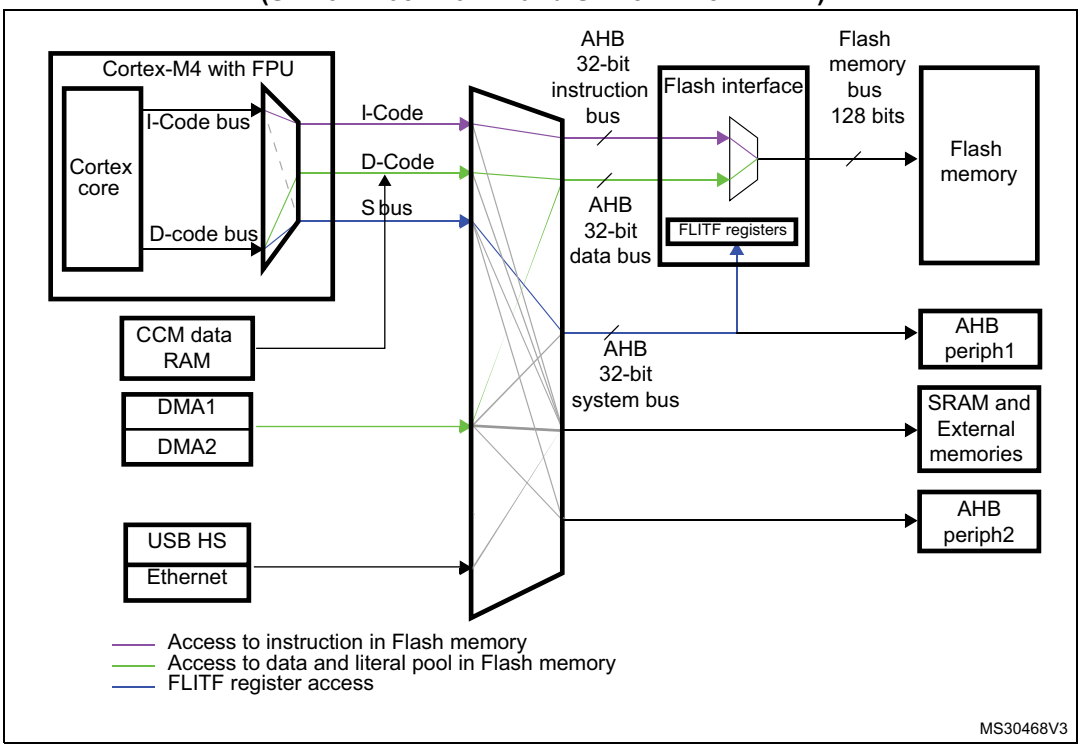
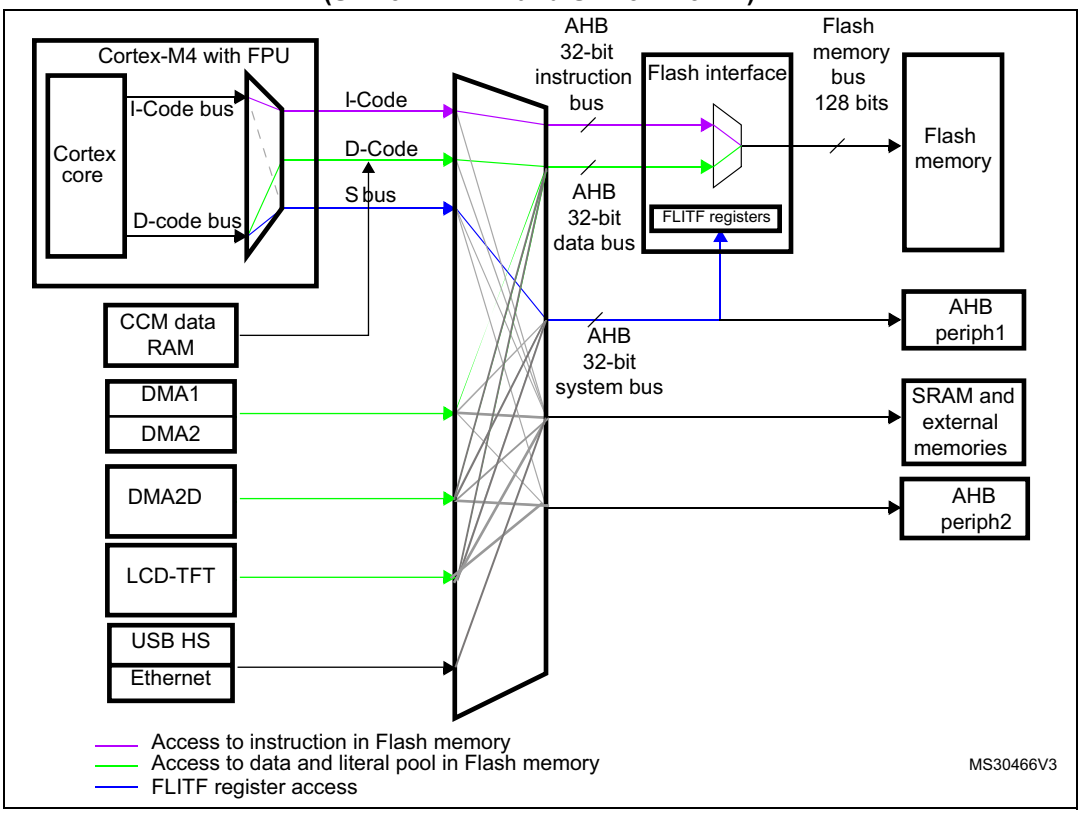


Figure 4. Flash memory interface connection inside system architecture (STM32F42xxx and STM32F43xxx)



3.3 Embedded Flash memory in STM32F405xx/07xx and STM32F415xx/17xx

The Flash memory has the following main features:

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

  The Flash memory is organized as follows:

  – A main memory block divided into 4 sectors of 16 Kbytes, 1 sector of 64 Kbytes, and 7 sectors of 128 Kbytes

  – System memory from which the device boots in System memory boot mode

  – 512 OTP (one-time programmable) bytes for user data

  The OTP area contains 16 additional bytes used to lock the corresponding OTP data block.

  – Option bytes to configure read and write protection, BOR level, watchdog software/hardware and reset when the device is in Standby or Stop mode.
- Low-power modes (for details refer to the Power control (PWR) section of the reference manual)

Table 5. Flash module organization (STM32F40x and STM32F41x)

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

3.4 Embedded Flash memory in STM32F42xxx and STM32F43xxx

The Flash memory has the following main features:

- Capacity up to 2 Mbyte with dual bank architecture supporting read-while-write capability (RWW)
- 128 bits wide data read
- Byte, half-word, word and double word write
- Sector, bank, and mass erase (both banks)
- Dual bank memory organization

  The Flash memory is organized as follows:

  – For each bank, a main memory block (1 Mbyte) divided into 4 sectors of 16 Kbytes, 1 sector of 64 Kbytes, and 7 sectors of 128 Kbytes

  – System memory from which the device boots in System memory boot mode

  – 512 OTP (one-time programmable) bytes for user data

  The OTP area contains 16 additional bytes used to lock the corresponding OTP data block.

  – Option bytes to configure read and write protection, BOR level, watchdog, dual bank boot mode, dual bank feature, software/hardware and reset when the device is in Standby or Stop mode.
- Dual bank organization on 1 Mbyte devices

  The dual bank feature on 1 Mbyte devices is enabled by setting the DB1M option bit.

  To obtain a dual bank Flash memory, the last 512 Kbytes of the single bank (sectors [8:11]) are re-structured in the same way as the first 512 Kbytes.

  The sector numbering of dual bank memory organization is different from the single bank: the single bank memory contains 12 sectors whereas the dual bank memory contains 16 sectors (see *Table 7: 1 Mbyte Flash memory single bank vs dual bank organization (STM32F42xxx and STM32F43xxx)*).

  For erase operation, the right sector numbering must be considered according the DB1M option bit.

  – When the DB1M bit is reset, the erase operation must be performed on the default sector number.

  – When the DB1M bit is set, to perform an erase operation on bank 2, the sector number must be programmed (sector number from 12 to 19). Refer to FLASH_CR register for SNB (Sector number) configuration.

Refer to *Table 8: 1 Mbyte single bank Flash memory organization (STM32F42xxx and STM32F43xxx)* and *Table 9: 1 Mbyte dual bank Flash memory organization (STM32F42xxx and STM32F43xxx)* for details on 1 Mbyte single bank and 1 Mbyte dual bank organizations.

Table 6. Flash module - 2 Mbyte dual bank organization (STM32F42xxx and STM32F43xxx)

Block	Bank	Name	Block base addresses	Size
Main memory	Bank 1	Sector 0	0x0800 0000 - 0x0800 3FFF	16 Kbytes
		Sector 1	0x0800 4000 - 0x0800 7FFF	16 Kbytes
		Sector 2	0x0800 8000 - 0x0800 BFFF	16 Kbytes
		Sector 3	0x0800 C000 - 0x0800 FFFF	16 Kbyte
		Sector 4	0x0801 0000 - 0x0801 FFFF	64 Kbytes
		Sector 5	0x0802 0000 - 0x0803 FFFF	128 Kbytes
		Sector 6	0x0804 0000 - 0x0805 FFFF	128 Kbytes
		-	-	-
		-	-	-
		-	-	-
		Sector 11	0x080E 0000 - 0x080F FFFF	128 Kbytes
	Bank 2	Sector 12	0x0810 0000 - 0x0810 3FFF	16 Kbytes
		Sector 13	0x0810 4000 - 0x0810 7FFF	16 Kbytes
		Sector 14	0x0810 8000 - 0x0810 BFFF	16 Kbytes
		Sector 15	0x0810 C000 - 0x0810 FFFF	16 Kbytes
		Sector 16	0x0811 0000 - 0x0811 FFFF	64 Kbytes
		Sector 17	0x0812 0000 - 0x0813 FFFF	128 Kbytes
		Sector 18	0x0814 0000 - 0x0815 FFFF	128 Kbytes
			-	-
			-	-
			-	-
		Sector 23	0x081E 0000 - 0x081F FFFF	128 Kbytes
System memory			0x1FFF 0000 - 0x1FFF 77FF	30 Kbytes
OTP			0x1FFF 7800 - 0x1FFF 7A0F	528 bytes
Option bytes	Bank 1		0x1FFF C000 - 0x1FFF C00F	16 bytes
	Bank 2		0x1FFE C000 - 0x1FFE C00F	16 bytes

Table 7. 1 Mbyte Flash memory single bank vs dual bank organization (STM32F42xxx and STM32F43xxx)

1 Mbyte single bank Flash memory (default)			1 Mbyte dual bank Flash memory		
DB1M=0			DB1M=1		
Main memory	Sector number	Sector size	Main memory	Sector number	Sector size
1MB	Sector 0	16 Kbytes	Bank 1 512KB	Sector 0	16 Kbytes
	Sector 1	16 Kbytes		Sector 1	16 Kbytes
	Sector 2	16 Kbytes		Sector 2	16 Kbytes
	Sector 3	16 Kbytes		Sector 3	16 Kbytes
	Sector 4	64 Kbytes		Sector 4	64 Kbytes
	Sector 5	128 Kbytes		Sector 5	128 Kbytes
	Sector 6	128 Kbytes		Sector 6	128 Kbytes
	Sector 7	128 Kbytes		Sector 7	128 Kbytes
	Sector 8	128 Kbytes	Bank 2 512KB	Sector 12	16 Kbytes
	Sector 9	128 Kbytes		Sector 13	16 Kbytes
	Sector 10	128 Kbytes		Sector 14	16 Kbytes
	Sector 11	128 Kbytes		Sector 15	16 Kbytes
	-	-		Sector 16	64 Kbytes
	-	-		Sector 17	128 Kbytes
	-	-		Sector 18	128 Kbytes
	-	-		Sector 19	128 Kbytes

Table 8. 1 Mbyte single bank Flash memory organization (STM32F42xxx and STM32F43xxx)

Block	Bank	Name	Block base addresses	Size
Main memory	Single bank	Sector 0	0x0800 0000 - 0x0800 3FFF	16 Kbytes
		Sector 1	0x0800 4000 - 0x0800 7FFF	16 Kbytes
		Sector 2	0x0800 8000 - 0x0800 BFFF	16 Kbytes
		Sector 3	0x0800 C000 - 0x0800 FFFF	16 Kbytes
		Sector 4	0x0801 0000 - 0x0801 FFFF	64 Kbytes
		Sector 5	0x0802 0000 - 0x0803 FFFF	128 Kbytes
		Sector 6	0x0804 0000 - 0x0805 FFFF	128 Kbytes
		Sector 7	0x0806 0000 - 0x0807 FFFF	128 Kbytes
		Sector 8	0x0808 0000 - 0x0809 FFFF	128 Kbytes
		Sector 9	0x080A 0000 - 0x080B FFFF	128 Kbytes
		Sector 10	0x080C 0000 - 0x080D FFFF	128 Kbytes
		Sector 11	0x080E 0000 - 0x080F FFFF	128 Kbytes

Table 8. 1 Mbyte single bank Flash memory organization (STM32F42xxx and STM32F43xxx) (continued)

Block	Bank	Name	Block base addresses	Size
System memory			0x1FFF 0000 - 0x1FFF 77FF	30 Kbytes
OTP			0x1FFF 7800 - 0x1FFF 7A0F	528 bytes
Option bytes			0x1FFF C000 - 0x1FFF C00F	16 bytes
			0x1FFE C000 - 0x1FFE C00F	16 bytes

Table 9. 1 Mbyte dual bank Flash memory organization (STM32F42xxx and STM32F43xxx)

Block		Name	Block base addresses	Size
Main memory	Bank 1	Sector 0	0x0800 0000 - 0x0800 3FFF	16 Kbytes
		Sector 1	0x0800 4000 - 0x0800 7FFF	16 Kbytes
		Sector 2	0x0800 8000 - 0x0800 BFFF	16 Kbytes
		Sector 3	0x0800 C000 - 0x0800 FFFF	16 Kbyte
		Sector 4	0x0801 0000 - 0x0801 FFFF	64 Kbytes
		Sector 5	0x0802 0000 - 0x0803 FFFF	128 Kbytes
		Sector 6	0x0804 0000 - 0x0805 FFFF	128 Kbytes
		Sector 7	0x0806 0000 - 0x0807 FFFF	128 Kbytes
	Bank 2	Sector 12	0x0808 0000 - 0x0808 3FFF	16 Kbytes
		Sector 13	0x0808 4000 - 0x0808 7FFF	16 Kbytes
		Sector 14	0x0808 0000 - 0x0808 BFFF	16 Kbytes
		Sector 15	0x0808 C000 - 0x0808 FFFF	16 Kbytes
		Sector 16	0x0809 0000 - 0x0809 FFFF	64 Kbytes
		Sector 17	0x080A 0000 - 0x080B FFFF	128 Kbytes
		Sector 18	0x080C 0000 - 0x080D FFFF	128 Kbytes
		Sector 19	0x080E 0000 - 0x080F FFFF	128 Kbytes
System memory			0x1FFF 0000 - 0x1FFF 77FF	30 Kbytes
OTP			0x1FFF 7800 - 0x1FFF 7A0F	528 bytes
Option bytes	Bank 1		0x1FFF C000 - 0x1FFF C00F	16 bytes
	Bank 2		0x1FFE C000 - 0x1FFE C00F	16 bytes

3.5 Read interface

3.5.1 Relation between CPU clock frequency and Flash memory read time

To correctly read data from Flash memory, the number of wait states (LATENCY) must be correctly programmed in the Flash access control register (FLASH_ACR) according to the frequency of the CPU clock (HCLK) and the supply voltage of the device.

The prefetch buffer must be disabled when the supply voltage is below 2.1 V. The correspondence between wait states and CPU clock frequency is given in *Table 10* and *Table 11*.

Note: On STM32F405xx/07xx and STM32F415xx/17xx devices:

- when VOS = '0', the maximum value of f_{HCLK} = 144 MHz.

- when VOS = '1', the maximum value of f_{HCLK} = 168 MHz.

On STM32F42xxx and STM32F43xxx devices:

- when VOS[1:0] = '0x01', the maximum value of f_{HCLK} is 120 MHz.

- when VOS[1:0] = '0x10', the maximum value of f_{HCLK} is 144 MHz. It can be extended to 168 MHz by activating the over-drive mode.

- when VOS[1:0] = '0x11, the maximum value of f_{HCLK} is 168 MHz. It can be extended to 180 MHz by activating the over-drive mode.

- The over-drive mode is not available when V_{DD} ranges from 1.8 to 2.1 V.

Refer to Section 5.1.4: Voltage regulator for STM32F42xxx and STM32F43xxx for details on how to activate the over-drive mode.

Table 10. Number of wait states according to CPU clock (HCLK) frequency (STM32F405xx/07xx and STM32F415xx/17xx)

Wait states (WS) (LATENCY)	HCLK (MHz)			
	Voltage range 2.7 V - 3.6 V	Voltage range 2.4 V - 2.7 V	Voltage range 2.1 V - 2.4 V	Voltage range 1.8 V - 2.1 V Prefetch OFF
0 WS (1 CPU cycle)	0 < HCLK ≤30	0 < HCLK ≤24	0 < HCLK ≤22	0 < HCLK ≤20
1 WS (2 CPU cycles)	30 < HCLK ≤60	24 < HCLK ≤48	22 < HCLK ≤44	20 <HCLK ≤40
2 WS (3 CPU cycles)	60 < HCLK ≤90	48 < HCLK ≤72	44 < HCLK ≤66	40 < HCLK ≤60
3 WS (4 CPU cycles)	90 < HCLK ≤120	72 < HCLK ≤96	66 < HCLK ≤88	60 < HCLK ≤80
4 WS (5 CPU cycles)	120 < HCLK ≤150	96 < HCLK ≤120	88 < HCLK ≤110	80 < HCLK ≤100
5 WS (6 CPU cycles)	150 < HCLK ≤168	120 < HCLK ≤144	110 < HCLK ≤132	100 < HCLK ≤120
6 WS (7 CPU cycles)		144 < HCLK ≤168	132 < HCLK ≤154	120 < HCLK ≤140
7 WS (8 CPU cycles)			154 < HCLK ≤168	140 < HCLK ≤160

Table 11. Number of wait states according to CPU clock (HCLK) frequency (STM32F42xxx and STM32F43xxx)

Wait states (WS) (LATENCY)	HCLK (MHz)			
	Voltage range 2.7 V - 3.6 V	Voltage range 2.4 V - 2.7 V	Voltage range 2.1 V - 2.4 V	Voltage range 1.8 V - 2.1 V Prefetch OFF
0 WS (1 CPU cycle)	0 <HCLK ≤30	0 <HCLK ≤24	0 <HCLK ≤22	0 < HCLK ≤20
1 WS (2 CPU cycles)	30 <HCLK ≤60	24 < HCLK ≤48	22 <HCLK ≤44	20 <HCLK ≤40
2 WS (3 CPU cycles)	60 <HCLK ≤90	48 < HCLK ≤72	44 < HCLK ≤66	40 < HCLK ≤60
3 WS (4 CPU cycles)	90 <HCLK ≤120	72 < HCLK ≤96	66 <HCLK ≤88	60 < HCLK ≤80
4 WS (5 CPU cycles)	120 <HCLK ≤150	96 < HCLK ≤120	88 < HCLK ≤110	80 < HCLK ≤100
5 WS (6 CPU cycles)	150 <HCLK ≤180	120 <HCLK ≤144	110 < HCLK ≤132	100 < HCLK ≤120
6 WS (7 CPU cycles)		144 <HCLK ≤168	132 < HCLK ≤154	120 < HCLK ≤140
7 WS (8 CPU cycles)		168 <HCLK ≤180	154 <HCLK ≤176	140 < HCLK ≤160
8 WS (9 CPU cycles)			176 <HCLK ≤180	160 < HCLK ≤168

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

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

Increasing the CPU frequency

1. Program the new number of wait states to the LATENCY bits in the FLASH_ACR register
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_ACR
5. Check that the new number of wait states is used to access the Flash memory by reading the FLASH_ACR register

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

3.5.2 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.

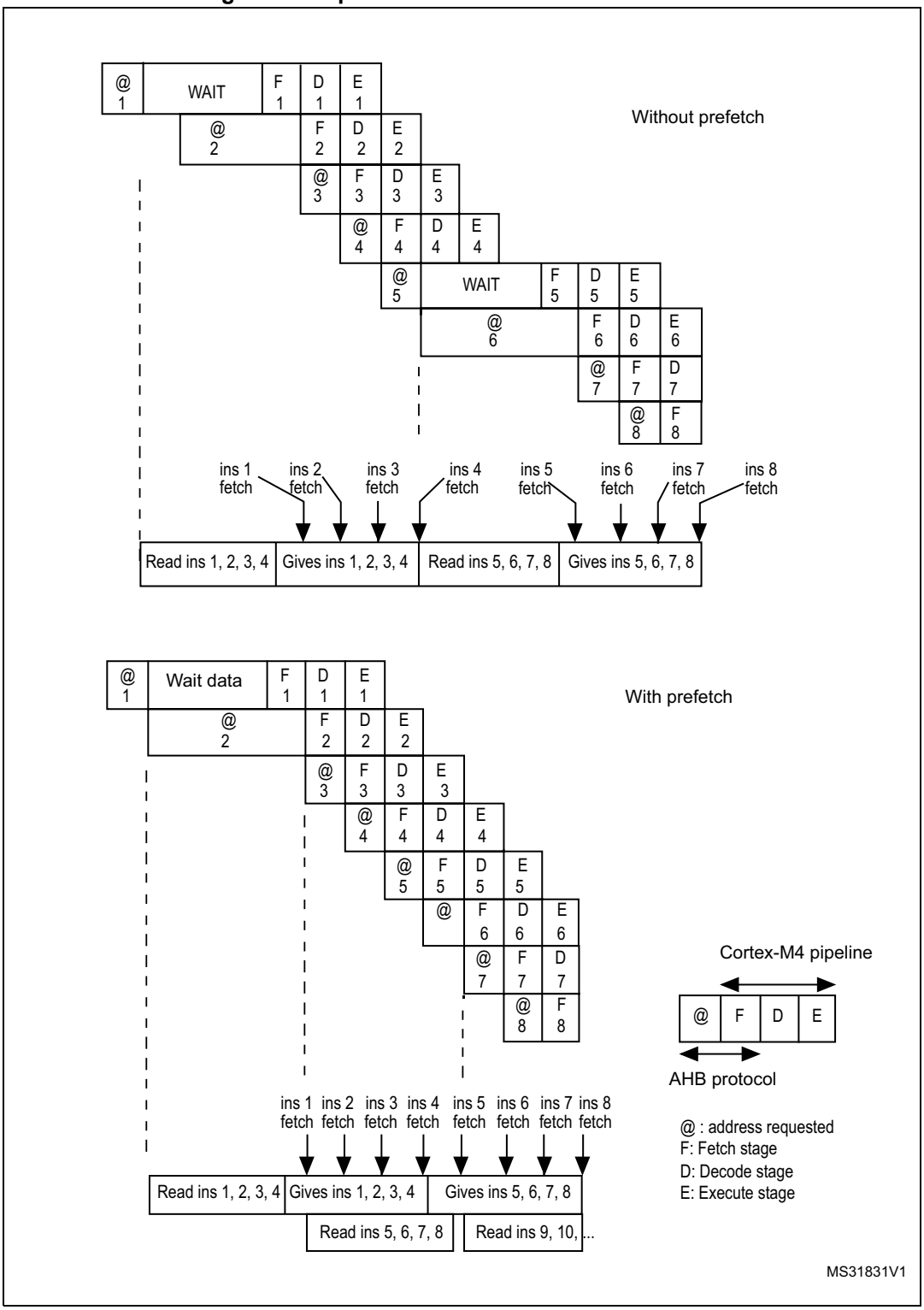
To release the processor full performance, the accelerator implements an instruction prefetch queue and branch cache which increases program execution speed from the 128-bit Flash memory. Based on CoreMark benchmark, the performance achieved thanks to the ART accelerator is equivalent to 0 wait state program execution from Flash memory at a CPU frequency up to 180 MHz.

Instruction prefetch

Each Flash memory read operation provides 128 bits from either four instructions of 32 bits or 8 instructions of 16 bits according to the program launched. So, in case of sequential code, at least four CPU cycles are needed to execute the previous read instruction line. Prefetch on the I-Code 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_ACR register. This feature is useful if at least one wait state is needed to access the Flash memory.

Figure 5 shows the execution of sequential 32-bit instructions with and without prefetch when 3 WSs are needed to access the Flash memory.

Figure 5. Sequential 32-bit instruction execution



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

Instruction cache memory

To limit the time lost due to jumps, it is possible to retain 64 lines of 128 bits in an instruction cache memory. This feature can be enabled by setting the instruction cache enable (ICEN) bit in the FLASH_ACR register. 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.

Data management

Literal pools are fetched from Flash memory through the D-Code bus during the execution stage of the CPU pipeline. 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 D-Code have priority over accesses through the AHB instruction bus I-Code.

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_ACR register. This feature works like the instruction cache memory, but the retained data size is limited to 8 rows of 128 bits.

Note: Data in user configuration sector are not cacheable.

3.6 Erase and program operations

For any Flash memory program operation (erase or program), the CPU clock frequency (HCLK) must be at least 1 MHz. The contents of the Flash memory are not guaranteed if a device reset occurs during a Flash memory operation.

Any attempt to read the Flash memory on STM32F4xx while it is being written or erased, causes the bus to stall. Read operations are processed correctly once the program operation has completed. This means that code or data fetches cannot be performed while a write/erase operation is ongoing.

On STM32F42xxx and STM32F43xxx devices, two banks are available allowing read operation from one bank while a write/erase operation is performed to the other bank.

3.6.1 Unlocking the Flash control register

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 key register (FLASH_KEYR)

Any wrong sequence will return a bus error and lock up the FLASH_CR register until the next reset.

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 is not accessible in write mode when the BSY bit in the FLASH_SR register 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.

3.6.2 Program/erase parallelism

The Parallelism size is configured through the PSIZE field in the FLASH_CR register. It represents the number of bytes to be programmed each time a write operation occurs to the Flash memory. PSIZE is limited by the supply voltage and by whether the external V_{PP} supply is used or not. It must therefore be correctly configured in the FLASH_CR register before any programming/erasing operation.

A Flash memory erase operation can only be performed by sector, bank or for the whole Flash memory (mass erase). The erase time depends on PSIZE programmed value. For more details on the erase time, refer to the electrical characteristics section of the device datasheet.

Table 12 provides the correct PSIZE values.

Table 12. Program/erase parallelism

	Voltage range 2.7 - 3.6 V with External V_{PP}	Voltage range 2.7 - 3.6 V	Voltage range 2.4 - 2.7 V	Voltage range 2.1 - 2.4 V	Voltage range 1.8 V - 2.1 V
Parallelism size	x64	x32	x16		x8
PSIZE(1:0)	11	10	01		00

Note: Any program or erase operation started with inconsistent program parallelism/voltage range settings may lead to unpredicted results. Even if a subsequent read operation indicates that the logical value was effectively written to the memory, this value may not be retained.

To use V_{PP}, an external high-voltage supply (between 8 and 9 V) must be applied to the V_{PP} pad. The external supply must be able to sustain this voltage range even if the DC consumption exceeds 10 mA. It is advised to limit the use of VPP to initial programming on the factory line. The V_{PP} supply must not be applied for more than an hour, otherwise the Flash memory might be damaged.

3.6.3 Erase

The Flash memory erase operation can be performed at sector level or on the whole Flash memory (Mass Erase). Mass Erase does not affect the OTP sector or the configuration sector.

Sector Erase

To erase a sector, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH_SR register
2. Set the SER bit and select the sector out of the 12 sectors (for STM32F405xx/07xx and STM32F415xx/17xx) and out of 24 (for STM32F42xxx and STM32F43xxx) in the main memory block you wish to erase (SNB) in the FLASH_CR register
3. Set the STRT bit in the FLASH_CR register
4. Wait for the BSY bit to be cleared

Bank erase in STM32F42xxx and STM32F43xxx devices

To erase bank 1 or bank 2, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH_SR register
2. Set MER or MER1 bit accordingly in the FLASH_CR register
3. Set the STRT bit in the FLASH_CR register
4. Wait for the BSY bit to be reset.

Mass Erase

To perform Mass Erase, the following sequence is recommended:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH_SR register
2. Set the MER bit in the FLASH_CR register (on STM32F405xx/07xx and STM32F415xx/17xx devices)
3. Set both the MER and MER1 bits in the FLASH_CR register (on STM32F42xxx and STM32F43xxx devices).
4. Set the STRT bit in the FLASH_CR register
5. Wait for the BSY bit to be cleared

Note: If MERx and SER bits are both set in the FLASH_CR register, mass erase is performed.

If both MERx and SER bits are reset and the STRT bit is set, an unpredictable behavior may occur without generating any error flag. This condition should be forbidden.

3.6.4 Programming

Standard programming

The Flash memory programming sequence is as follows:

1. Check that no main Flash memory operation is ongoing by checking the BSY bit in the FLASH_SR register.
2. Set the PG bit in the FLASH_CR register
3. Perform the data write operation(s) to the desired memory address (inside main memory block or OTP area):
   - Byte access in case of x8 parallelism
   - Half-word access in case of x16 parallelism
   - Word access in case of x32 parallelism
   - Double word access in case of x64 parallelism
4. Wait for the BSY bit to be cleared.

Note: Successive write operations are possible without the need of an erase operation when changing bits from ‘1’ to ‘0’. Writing ‘1’ requires a Flash memory erase operation.

If an erase and a program operation are requested simultaneously, the erase operation is performed first.

Programming errors

It is not allowed to program data to the Flash memory that would cross the 128-bit row boundary. In such a case, the write operation is not performed and a program alignment error flag (PGAERR) is set in the FLASH_SR register.

The write access type (byte, half-word, word or double word) must correspond to the type of parallelism chosen (x8, x16, x32 or x64). If not, the write operation is not performed and a program parallelism error flag (PGPERR) is set in the FLASH_SR register.

If the standard programming sequence is not respected (for example, if there is an attempt to write to a Flash memory address when the PG bit is not set), the operation is aborted and a program sequence error flag (PGSERR) is set in the FLASH_SR register.

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_CR register.

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

3.6.5 Read-while-write (RWW)

In STM32F42xxx and STM32F43xxx devices, the Flash memory is divided into two banks allowing read-while-write operations. This feature allows to perform a read operation from one bank while an erase or program operation is performed to the other bank.

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

Read from bank 1 while erasing bank 2

While executing a program code from bank 1, it is possible to perform an 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_SR register (BSY is active when erase/program operation is on going to bank 1 or bank 2)
2. Set MER or MER1 bit in the FLASH_CR register
3. Set the STRT bit in the FLASH_CR register
4. Wait for the BSY bit to be reset (or use the EOP interrupt).

Read from bank 1 while programming bank 2

While executing a program code (over the I-Code bus) from bank 1,it is possible to perform an program operation to 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_SR register (BSY is active when erase/program operation is on going on bank 1 or bank 2)
2. Set the PG bit in the FLASH_CR register
3. Perform the data write operation(s) to the desired memory address inside main memory block or OTP area
4. Wait for the BSY bit to be reset.

3.6.6 Interrupts

Setting the end of operation interrupt enable bit (EOPIE) in the FLASH_CR register enables interrupt generation when an erase or program operation ends, that is when the busy bit (BSY) in the FLASH_SR register is cleared (operation completed, correctly or not). In this case, the end of operation (EOP) bit in the FLASH_SR register is set.

If an error occurs during a program, an erase, or a read operation request, one of the following error flags is set in the FLASH_SR register:

- PGAERR, PGPERR, PGSERR (Program error flags)
- WRPERR (Protection error flag)
- RDERR (Read protection error flag) for STM32F42xxx and STM32F43xxx devices only.

In this case, if the error interrupt enable bit (ERRIE) is set in the FLASH_CR register, an interrupt is generated and the operation error bit (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.

Table 13. Flash interrupt request

Interrupt event	Event flag	Enable control bit
End of operation	EOP	EOPIE
Write protection error	WRPERR	ERRIE
Programming error	PGAERR, PGPERR, PGSERR	ERRIE
Read protection error	RDERR	ERRIE

3.7 Option bytes

3.7.1 Description of user option bytes

The option bytes are configured by the end user depending on the application requirements. *Table 14* shows the organization of these bytes inside the user configuration sector.

Table 14. Option byte organization

Address	[63:16]	[15:0]
0x1FFF C000	Reserved	ROP & user option bytes (**RDP** & **USER**)
0x1FFF C008	Reserved	SPRMOD and Write protection **nWRP bits for sectors 0 to 11**

Table 14. Option byte organization (continued)

Address	[63:16]	[15:0]
0x1FFE C000	Reserved	Reserved
0x1FFE C008	Reserved	SPRMOD and Write protection **nWRP bits for sectors 12 to 23**

Table 15. Description of the option bytes (STM32F405xx/07xx and STM32F415xx/17xx)

Option bytes (word, address 0x1FFF C000)	
RDP: *Read protection option byte.* The read protection is used to protect the software code stored in Flash memory.	
Bits 15:8	0xAA: Level 0, no protection 0xCC: Level 2, chip protection (debug and boot from RAM features disabled) Others: Level 1, read protection of memories (debug features limited)
USER: User option byte This byte is used to configure the following features: – Select the watchdog event: Hardware or software – Reset event when entering the Stop mode – Reset event when entering the Standby mode	
Bit 7	**nRST_STDBY** 0: Reset generated when entering the Standby mode 1: No reset generated
Bit 6	**nRST_STOP** 0: Reset generated when entering the Stop mode 1: No reset generated
Bit 5	**WDG_SW** 0: Hardware independent watchdog 1: Software independent watchdog
Bit 4	0x1: Not used
Bits 3:2	**BOR_LEV:** BOR reset Level These bits contain the supply level threshold that activates/releases the reset. They can be written to program a new BOR level value into Flash memory. 00: BOR Level 3 (VBOR3), brownout threshold level 3 01: BOR Level 2 (VBOR2), brownout threshold level 2 10: BOR Level 1 (VBOR1), brownout threshold level 1 11: BOR off, POR/PDR reset threshold level is applied *Note: For full details on BOR characteristics, refer to the "Electrical characteristics" section of the product datasheet.*
Bits 1:0	0x1: Not used

Table 15. Description of the option bytes (STM32F405xx/07xx and STM32F415xx/17xx) (continued)

Option bytes (word, address 0x1FFF C008)	
Bits 15:12	0xF: Not used
nWRP: *Flash memory write protection option bytes* Sectors 0 to 11 can be write protected.	
Bits 11:0	**nWRPi** 0: Write protection active on selected sector 1: Write protection not active on selected sector

Table 16. Description of the option bytes (STM32F42xxx and STM32F43xxx)

Option bytes (word, address 0x1FFF C000)	
RDP: *Read protection option byte.* The read protection is used to protect the software code stored in Flash memory.	
Bit 15:8	0xAA: Level 0, no protection 0xCC: Level 2, chip protection (debug and boot from RAM features disabled) Others: Level 1, read protection of memories (debug features limited)
USER: User option byte This byte is used to configure the following features: Select the watchdog event: Hardware or software Reset event when entering the Stop mode Reset event when entering the Standby mode	
Bit 7	**nRST_STDBY** 0: Reset generated when entering the Standby mode 1: No reset generated
Bit 6	**nRST_STOP** 0: Reset generated when entering the Stop mode 1: No reset generated
Bit 5	**WDG_SW** 0: Hardware independent watchdog 1: Software independent watchdog
Bit 4	**BFB2**: Dual bank boot 0: Boot from Flash memory bank 1 or system memory depending on boot pin state (Default). 1: Boot always from system memory (Dual bank boot mode).

Table 16. Description of the option bytes (STM32F42xxx and STM32F43xxx) (continued)

Bits 3:2	**BOR_LEV:** BOR reset Level These bits contain the supply level threshold that activates/releases the reset. They can be written to program a new BOR level value into Flash memory. 00: BOR Level 3 (VBOR3), brownout threshold level 3 01: BOR Level 2 (VBOR2), brownout threshold level 2 10: BOR Level 1 (VBOR1), brownout threshold level 1 11: BOR off, POR/PDR reset threshold level is applied *Note: For full details on BOR characteristics, refer to the "Electrical characteristics" section of the product datasheet.*
Bits 1:0	0x1: Not used
Option bytes (word, address 0x1FFF C008)	
Bit 15	**SPRMOD**: Selection of protection mode of nWPRi bits 0: nWPRi bits used for sector i write protection (Default) 1: nWPRi bits used for sector i PCROP protection (Sector)
Bit 14	**DB1M**: Dual bank 1 Mbyte Flash memory devices 0: 1 Mbyte single Flash memory (contiguous addresses in bank 1) 1: 1 Mbyte dual bank Flash memory. The Flash memory is organized as two banks of 512 Kbytes each (see *Table 7: 1 Mbyte Flash memory single bank vs dual bank organization (STM32F42xxx and STM32F43xxx)* and *Table 9: 1 Mbyte dual bank Flash memory organization (STM32F42xxx and STM32F43xxx)*). To perform an erase operation, the right sector must be programmed (see *Table 7* for information on the sector numbering scheme).
Bits 13:12	0x2: not used
nWRP: Flash memory write protection option bytes for bank 1. Sectors 0 to 11 can be write protected.	
Bits 11:0	**nWRPi:** If SPRMOD is reset (default value): 0: Write protection active on sector i. 1: Write protection not active on sector i. If SPRMOD is set (active): 0: PCROP protection not active on sector i. 1: PCROP protection active on sector i.
Option bytes (word, address 0x1FFE C000)	
Bit 15:0	0xFFFF: not used

Table 16. Description of the option bytes (STM32F42xxx and STM32F43xxx) (continued)

Option bytes (word, address 0x1FFE C008)	
Bit 15:12	0xF: not used
nWRP: Flash memory write protection option bytes for bank 2. Sectors 12 to 23 can be write protected.	
Bits 11: 0	nWRPi: If SPRMOD is reset (default value): 0: Write protection active on sector i. 1: Write protection not active on sector i. If SPRMOD is set (active): 0: PCROP protection not active on sector i. 1: PCROP protection active on sector i.

3.7.2 Programming user option bytes

To run any operation on this sector, the option lock bit (OPTLOCK) in the Flash option control register (FLASH_OPTCR) must be cleared. To be allowed to clear this bit, you have to perform the following sequence:

1. Write OPTKEY1 = 0x0819 2A3B in the Flash option key register (FLASH_OPTKEYR)
2. Write OPTKEY2 = 0x4C5D 6E7F in the Flash option key register (FLASH_OPTKEYR)

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

Modifying user option bytes on STM32F405xx/07xx and STM32F415xx/17xx

To modify the user option value, follow the sequence below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH_SR register
2. Write the desired option value in the FLASH_OPTCR register.
3. Set the option start bit (OPTSTRT) in the FLASH_OPTCR register
4. Wait for the BSY bit to be cleared.

Note: The value of an option is automatically modified by first erasing the user configuration sector and then programming all the option bytes with the values contained in the FLASH_OPTCR register.

Modifying user option bytes on STM32F42xxx and STM32F43xxx

The user option bytes for bank 1 and bank 2 cannot be modified independently. They must be updated concurrently.

To modify the user option byte value, follow the sequence below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH_SR register
2. Write the bank 2 option byte value in the FLASH_OPTCR1 register
3. Write the bank 1 option byte value in the FLASH_OPTCR register.
4. Set the option start bit (OPTSTRT) in the FLASH_OPTCR register
5. Wait for the BSY bit to be cleared

Note: The value of an option byte is automatically modified by first erasing the user configuration sector (bank 1 and 2) and then programming all the option bytes with the values contained in the FLASH_OPTCR and FLASH_OPTCR1 registers.

3.7.3 Read protection (RDP)

The user area in the Flash memory can be protected against read operations by an entrusted code. Three read protection levels are defined:

- Level 0: no read protection

  When the read protection level is set to Level 0 by writing 0xAA into the read protection option byte (RDP), all read/write operations (if no write protection is set) from/to the Flash memory or the backup SRAM are possible in all boot configurations (Flash user boot, debug or boot from RAM).

- Level 1: read protection enabled

  It is the default read protection level after option byte erase. The read protection Level 1 is activated by writing any value (except for 0xAA and 0xCC used to set Level 0 and Level 2, respectively) into the RDP option byte. When the read protection Level 1 is set:

  – No access (read, erase, program) to Flash memory or backup SRAM can be performed while the debug feature is connected or while booting from RAM or system memory bootloader. A bus error is generated in case of read request.

  – When booting from Flash memory, accesses (read, erase, program) to Flash memory and backup SRAM from user code are allowed.

  When Level 1 is active, programming the protection option byte (RDP) to Level 0 causes the Flash memory and the backup SRAM to be mass-erased. As a result the user code area is cleared before the read protection is removed. The mass erase only erases the user code area. The other option bytes including write protections remain unchanged from before the mass-erase operation. The OTP area is not affected by mass erase and remains unchanged. 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.

- Level 2: debug/chip read protection disabled

  The read protection Level 2 is activated by writing 0xCC to the RDP option byte. When the read protection Level 2 is set:

  – All protections provided by Level 1 are active.

  – Booting from RAM or system memory bootloader is no more allowed.

  – JTAG, SWV (single-wire viewer), ETM, and boundary scan are disabled.

  – User option bytes can no longer be changed.

  – When booting from Flash memory, accesses (read, erase and program) to Flash memory and backup SRAM from user code are allowed.

  Memory read protection Level 2 is an irreversible operation. When Level 2 is activated, the level of protection cannot be decreased to Level 0 or Level 1.

Note: The JTAG port is permanently disabled when Level 2 is active (acting as a JTAG fuse). As a consequence, boundary scan cannot be performed. STMicroelectronics is not able to perform analysis on defective parts on which the Level 2 protection has been set.

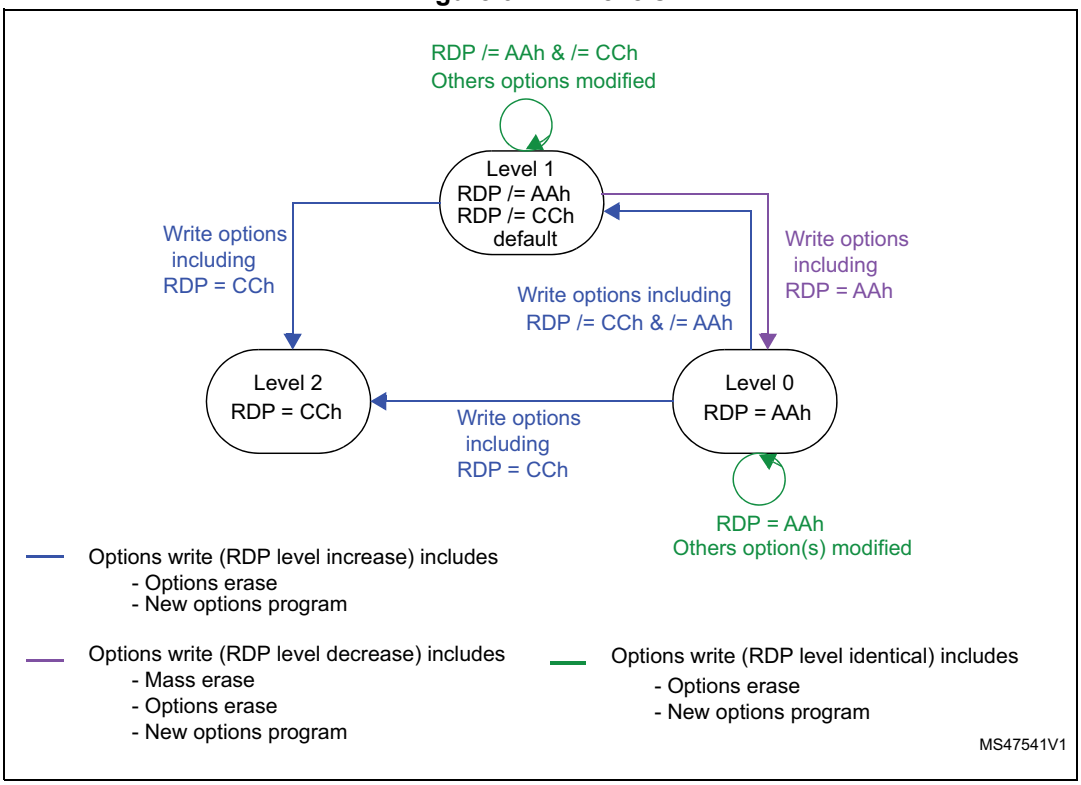
Table 17. Access versus read protection level

Memory area	Protection Level	Debug features, Boot from RAM or from System memory bootloader			Booting from Flash memory		
		Read	Write	Erase	Read	Write	Erase
Main Flash Memory and Backup SRAM	Level 1	NO		NO(1)	YES		
	Level 2	NO			YES		
Option Bytes	Level 1	YES			YES		
	Level 2	NO			NO		
OTP	Level 1	NO		NA	YES		NA
	Level 2	NO		NA	YES		NA

1. The main Flash memory and backup SRAM are only erased when the RDP changes from level 1 to 0. The OTP area remains unchanged.

Figure 6 shows how to go from one RDP level to another.

Figure 6. RDP levels



3.7.4 Write protections

Up to 24 user sectors in Flash memory can be protected against unwanted write operations due to loss of program counter contexts. When the non-write protection nWRPi bit (0 ≤ i ≤11) in the FLASH_OPTCR or FLASH_OPTCR1 registers is low, the corresponding sector

cannot be erased or programmed. Consequently, a mass erase cannot be performed if one of the sectors is write-protected.

If an erase/program operation to a write-protected part of the Flash memory is attempted (sector protected by write protection bit, OTP part locked or part of the Flash memory that can never be written like the ICP), the write protection error flag (WRPERR) is set in the FLASH_SR register.

On STM32F42xxx and STM32F43xxx devices, when the PCROP mode is set, the active level of nWRPi is high, and the corresponding sector i is write protected when nWRPi is high. A PCROP sector is automatically write protected.

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

Write protection error flag

If an erase/program operation to a write protected area of the Flash memory is performed, the Write Protection Error flag (WRPERR) is set in the FLASH_SR register.

If an erase operation is requested, the WRPERR bit is set when:

- Mass, bank, sector erase are configured (MER or MER/MER1 and SER = 1)
- A sector erase is requested and the Sector Number SNB field is not valid
- A mass erase is requested while at least one of the user sector is write protected by option bit (MER or MER/MER1 = 1 and nWRPi = 0 with 0 ≤i ≤11 bits in the FLASH_OPTCRx register
- A sector erase is requested on a write protected sector. (SER = 1, SNB = i and nWRPi = 0 with 0 ≤i ≤11 bits in the FLASH_OPTCRx register)
- The Flash memory is readout protected and an intrusion is detected.

If a program operation is requested, the WRPERR bit is set when:

- A write operation is performed on system memory or on the reserved part of the user specific sector.
- A write operation is performed to the user configuration sector
- A write operation is performed on a sector write protected by option bit.
- A write operation is requested on an OTP area which is already locked
- The Flash memory is read protected and an intrusion is detected.

3.7.5 Proprietary code readout protection (PCROP)

The proprietary readout protection (PCROP) is available only on STM32F42xxx and STM32F43xxx devices.

Flash memory user sectors (0 to 23) can be protected against D-bus read accesses by using the proprietary readout protection (PCROP).

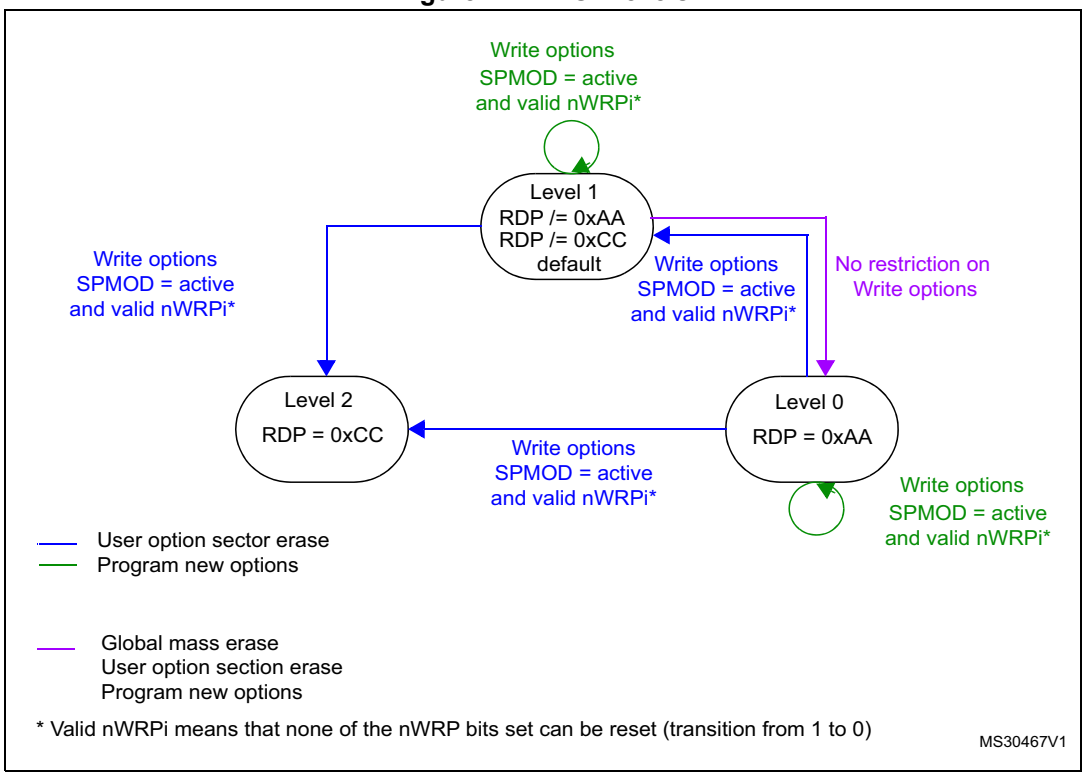
The PCROP protection is selected as follows, through the SPRMOD option bit in the FLASH_OPTCR register:

- SPRMOD = 0: nWRPi control the write protection of respective user sectors
- SPRMOD = 1: nWRPi control the read and write protection (PCROP) of respective user sectors.

When a sector is readout protected (PCROP mode activated), it can only be accessed for code fetch through ICODE Bus on Flash interface:

- Any read access performed through the D-bus triggers a RDERR flag error.
- Any program/erase operation on a PCROPed sector triggers a WRPERR flag error.

Figure 7. PCROP levels



The deactivation of the SPRMOD and/or the unprotection of PCROPed user sectors can only occur when, at the same time, the RDP level changes from 1 to 0. If this condition is not respected, the user option byte modification is cancelled and the write error WRPERR flag is set. The modification of the users option bytes (BOR_LEV, RST_STDBY, ..) is allowed since none of the active nWRPi bits is reset and SPRMOD is kept active.

Note: The active value of nWRPi bits is inverted when PCROP mode is active (SPRMOD =1).

If SPRMOD = 1 and nWRPi =1, then user sector i of bank 1, respectively bank 2 is read/write protected (PCROP).

3.8 One-time programmable bytes

Table 18 shows the organization of the one-time programmable (OTP) part of the OTP area.

Table 18. OTP area organization

Block	[128:96]	[95:64]	[63:32]	[31:0]	Address byte 0
0	OTP0	OTP0	OTP0	OTP0	0x1FFF 7800
	OTP0	OTP0	OTP0	OTP0	0x1FFF 7810
1	OTP1	OTP1	OTP1	OTP1	0x1FFF 7820
	OTP1	OTP1	OTP1	OTP1	0x1FFF 7830
. . .	. . .				. . .
15	OTP15	OTP15	OTP15	OTP15	0x1FFF 79E0
	OTP15	OTP15	OTP15	OTP15	0x1FFF 79F0
Lock block	LOCKB15 ... LOCKB12	LOCKB11 ... LOCKB8	LOCKB7 ... LOCKB4	LOCKB3 ... LOCKB0	0x1FFF 7A00

The OTP area is divided into 16 OTP data blocks of 32 bytes and one lock OTP block of 16 bytes. The OTP data and lock blocks cannot be erased. The lock block contains 16 bytes LOCKBi (0 ⩽ ≤15) to lock the corresponding OTP data block (blocks 0 to 15). Each OTP data block can be programmed until the value 0x00 is programmed in the corresponding OTP lock byte. The lock bytes must only contain 0x00 and 0xFF values, otherwise the OTP bytes might not be taken into account correctly.

3.9 Flash interface registers

3.9.1 Flash access control register (FLASH_ACR) for STM32F405xx/07xx and STM32F415xx/17xx

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

Address offset: 0x00
Reset value: 0x0000 0000

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

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

Bits 31:13 Reserved, must be kept cleared.

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

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

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

Bit 9 **ICEN:** Instruction cache enable
0: Instruction cache is disabled
1: Instruction cache is enabled

Bit 8 **PRFTEN:** Prefetch enable
0: Prefetch is disabled
1: Prefetch is enabled

Bits 7:3 Reserved, must be kept cleared.

Bits 2:0 **LATENCY[2:0]:** Latency
These bits represent the ratio of the CPU clock period to the Flash memory access time.
000: Zero wait state
001: One wait state
010: Two wait states
011: Three wait states
100: Four wait states
101: Five wait states
110: Six wait states
111: Seven wait states

3.9.2 Flash access control register (FLASH_ACR) for STM32F42xxx and STM32F43xxx

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

Address offset: 0x00
Reset value: 0x0000 0000

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

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

Bits 31:11 Reserved, must be kept cleared.

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

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

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

Bit 9 **ICEN:** Instruction cache enable
0: Instruction cache is disabled
1: Instruction cache is enabled

Bit 8 **PRFTEN:** Prefetch enable
0: Prefetch is disabled
1: Prefetch is enabled

Bits 7:4 Reserved, must be kept cleared.

Bits 3:0 **LATENCY[3:0]:** Latency
These bits represent the ratio of the CPU clock period to the Flash memory access time.
0000: Zero wait state
0001: One wait state
0010: Two wait states
...
1110: Fourteen wait states
1111: Fifteen wait states

3.9.3 Flash key register (FLASH_KEYR)

The Flash key register is used to allow access to the Flash control register and so, to allow program and erase operations.

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
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **FKEYR[31:0]**: FPEC key

The following values must be programmed consecutively to unlock the FLASH_CR register and allow programming/erasing it:

a) KEY1 = 0x45670123
b) KEY2 = 0xCDEF89AB

3.9.4 Flash option key register (FLASH_OPTKEYR)

The Flash option key register is used to allow program and erase operations in the user configuration sector.

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
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[31:0]**: Option byte key

The following values must be programmed consecutively to unlock the FLASH_OPTCR register and allow programming it:

a) OPTKEY1 = 0x08192A3B
b) OPTKEY2 = 0x4C5D6E7F

3.9.5 Flash status register (FLASH_SR) for STM32F405xx/07xx and STM32F415xx/17xx

The Flash status register gives information on ongoing program and erase operations.

Address offset: 0x0C
Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															BSY
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								PGSERR	PGPERR	PGAERR	WRPERR	Reserved		OPERR	EOP
								rc_w1	rc_w1	rc_w1	rc_w1			rc_w1	rc_w1

Bits 31:17 Reserved, must be kept cleared.

Bit 16 **BSY:** Busy
This bit indicates that a Flash memory operation is in progress. It is set at the beginning of a Flash memory operation and cleared when the operation finishes or an error occurs.
0: no Flash memory operation ongoing
1: Flash memory operation ongoing

Bits 15:8 Reserved, must be kept cleared.

Bit 7 **PGSERR:** Programming sequence error
Set by hardware when a write access to the Flash memory is performed by the code while the control register has not been correctly configured.
Cleared by writing 1.

Bit 6 **PGPERR:** Programming parallelism error
Set by hardware when the size of the access (byte, half-word, word, double word) during the program sequence does not correspond to the parallelism configuration PSIZE (x8, x16, x32, x64).
Cleared by writing 1.

Bit 5 **PGAERR:** Programming alignment error
Set by hardware when the data to program cannot be contained in the same 128-bit Flash memory row.
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 of the Flash memory.
Cleared by writing 1.

Bits 3:2 Reserved, must be kept cleared.

Bit 1 **OPERR:** Operation error

Set by hardware when a flash operation (programming / erase /read) request is detected and can not be run because of parallelism, alignment, or write protection error. This bit is set only if error interrupts are enabled (ERRIE = 1).

Bit 0 **EOP:** End of operation

Set by hardware when one or more Flash memory operations (program/erase) has/have completed successfully. It is set only if the end of operation interrupts are enabled (EOPIE = 1).

Cleared by writing a 1.

3.9.6 Flash status register (FLASH_SR) for STM32F42xxx and STM32F43xxx

The Flash status register gives information on ongoing program and erase operations.

Address offset: 0x0C

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															BSY
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							RDERR	PGSERR	PGPERR	PGAERR	WRPERR	Reserved		OPERR	EOP
							rc_w1	rc_w1	rc_w1	rc_w1	rc_w1			rc_w1	rc_w1

Bits 31:17 Reserved, must be kept cleared.

Bit 16 **BSY:** Busy

This bit indicates that a Flash memory operation is in progress to/from one bank. It is set at the beginning of a Flash memory operation and cleared when the operation finishes or an error occurs.

0: no Flash memory operation ongoing
1: Flash memory operation ongoing

Bits 15:9 Reserved, must be kept cleared.

Bit 8 **RDERR:** Proprietary readout protection (PCROP) error

Set by hardware when a read access through the D-bus is performed to an address belonging to a proprietary readout protected Flash sector.

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 the control register has not been correctly configured.

Cleared by writing 1.

Bit 6 **PGPERR:** Programming parallelism error
Set by hardware when the size of the access (byte, half-word, word, double word) during the program sequence does not correspond to the parallelism configuration PSIZE (x8, x16, x32, x64).
Cleared by writing 1.

Bit 5 **PGAERR:** Programming alignment error
Set by hardware when the data to program cannot be contained in the same 128-bit Flash memory row.
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 of the Flash memory.
Cleared by writing 1.

Bits 3:2 Reserved, must be kept cleared.

Bit 1 **OPERR:** Operation error
Set by hardware when a flash operation (programming/erase/read) request is detected and can not be run because of parallelism, alignment, write or read (PCROP) protection error. This bit is set only if error interrupts are enabled (ERRIE = 1).

Bit 0 **EOP:** End of operation
Set by hardware when one or more Flash memory operations (program/erase) has/have completed successfully. It is set only if the end of operation interrupts are enabled (EOPIE = 1).
Cleared by writing a 1.

3.9.7 Flash control register (FLASH_CR) for STM32F405xx/07xx and STM32F415xx/17xx

The Flash control register is used to configure and start Flash memory operations.

Address offset: 0x10

Reset value: 0x8000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Reserved					ERRIE	EOPIE	Reserved							STRT
rs						rw	rw								rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						PSIZE[1:0]		Res.	SNB[3:0]				MER	SER	PG
						rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK:** Lock

Write to 1 only. When it is set, this bit indicates that the FLASH_CR register is locked. It is cleared by hardware after detecting the unlock sequence.
In the event of an unsuccessful unlock operation, this bit remains set until the next reset.

Bits 30:26 Reserved, must be kept cleared.

Bit 25 **ERRIE:** Error interrupt enable

This bit enables the interrupt generation when the OPERR bit in the FLASH_SR register is set to 1.

0: Error interrupt generation disabled
1: Error interrupt generation enabled

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

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

0: Interrupt generation disabled
1: Interrupt generation enabled

Bits 23:17 Reserved, must be kept cleared.

Bit 16 **STRT:** Start

This bit triggers an erase operation when set. It is set only by software and cleared when the BSY bit is cleared.

Bits 15:10 Reserved, must be kept cleared.

Bits 9:8 **PSIZE[1:0]:** Program size

These bits select the program parallelism.

00 program x8
01 program x16
10 program x32
11 program x64

Bit 7 Reserved, must be kept cleared.

Bits 6:3 **SNB[3:0]:** Sector number

These bits select the sector to erase.

0000 sector 0
0001 sector 1
...
1011 sector 11
Others not allowed

Bit 2 **MER:** Mass Erase

Erase activated for all user sectors.

Bit 1 **SER:** Sector Erase

Sector Erase activated.

Bit 0 **PG:** Programming

Flash programming activated.

3.9.8 Flash control register (FLASH_CR) for STM32F42xxx and STM32F43xxx

The Flash control register is used to configure and start Flash memory operations.

Address offset: 0x10

Reset value: 0x8000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Reserved					ERRIE	EOPIE	Reserved							STRT
rs						rw	rw								rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MER1	Reserved					PSIZE[1:0]		SNB[4:0]					MER	SER	PG
rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK:** Lock

Write to 1 only. When it is set, this bit indicates that the FLASH_CR register is locked. It is cleared by hardware after detecting the unlock sequence.
In the event of an unsuccessful unlock operation, this bit remains set until the next reset.

Bits 30:26 Reserved, must be kept cleared.

Bit 25 **ERRIE:** Error interrupt enable

This bit enables the interrupt generation when the OPERR bit in the FLASH_SR register is set to 1.

0: Error interrupt generation disabled
1: Error interrupt generation enabled

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

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

0: Interrupt generation disabled
1: Interrupt generation enabled

Bits 23:17 Reserved, must be kept cleared.

Bit 16 **STRT:** Start

This bit triggers an erase operation when set. It is set only by software and cleared when the BSY bit is cleared.

Bit 15 **MER1:** Mass Erase of bank 2 sectors

Erase activated for bank 2 user sectors 12 to 23.

Bits 14:10 Reserved, must be kept cleared.

Bits 9:8 **PSIZE[1:0]:** Program size

These bits select the program parallelism.

00 program x8
01 program x16
10 program x32
11 program x64

Bits 7:3 **SNB[3:0]:** Sector number
These bits select the sector to erase.
0000: sector 0
0001: sector 1
...
01011: sector 11
01100: not allowed
01101: not allowed
01110: not allowed
01111: not allowed
10000: section 12
10001: section 13
...
11011 sector 23
11100: not allowed
11101: not allowed
11110: not allowed
11111: not allowed

Bit 2 **MER:** Mass Erase of bank 1 sectors
Erase activated of bank 1 sectors.

Bit 1 **SER:** Sector Erase
Sector Erase activated.

Bit 0 **PG:** Programming
Flash programming activated.

3.9.9 Flash option control register (FLASH_OPTCR) for STM32F405xx/07xx and STM32F415xx/17xx

The FLASH_OPTCR register is used to modify the user option bytes.

Address offset: 0x14

Reset value: 0x0FFF AAED. The option bits are loaded with values from Flash memory at reset release.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				nWRP[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
RDP[7:0]								nRST_STDBY	nRST_STOP	WDG_SW	Reserved	BOR_LEV		OPTSTRT	OPTLOCK
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rs	rs

Bits 31:28 Reserved, must be kept cleared.

Bits 27:16 **nWRP[11:0]:** Not write protect

These bits contain the value of the write-protection option bytes after reset. They can be written to program a new write protect value into Flash memory.

0: Write protection active on selected sector
1: Write protection inactive on selected sector

Bits 15:8 **RDP[7:0]:** Read protect

These bits contain the value of the read-protection option level after reset. They can be written to program a new read protection value into Flash memory.

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

Bits 7:5 **USER[2:0]:** User option bytes

These bits contain the value of the user option byte after reset. They can be written to program a new user option byte value into Flash memory.

Bit 7: nRST_STDBY
Bit 6: nRST_STOP
Bit 5: WDG_SW

Note: When changing the WDG mode from hardware to software or from software to hardware, a system reset is required to make the change effective.

Bit 4 Reserved, must be kept cleared. Always read as “0”.

Bits 3:2 **BOR_LEV[1:0]:** BOR reset Level

These bits contain the supply level threshold that activates/releases the reset. They can be written to program a new BOR level. By default, BOR is off. When the supply voltage (V_{DD}) drops below the selected BOR level, a device reset is generated.

00: BOR Level 3 (VBOR3), brownout threshold level 3
01: BOR Level 2 (VBOR2), brownout threshold level 2
10: BOR Level 1 (VBOR1), brownout threshold level 1
11: BOR off, POR/PDR reset threshold level is applied

Note: For full details about BOR characteristics, refer to the “Electrical characteristics” section in the device datasheet.

Bit 1 **OPTSTRT:** Option start

This bit triggers a user option operation when set. It is set only by software and cleared when the BSY bit is cleared.

Bit 0 **OPTLOCK:** Option lock

Write to 1 only. When this bit is set, it indicates that the FLASH_OPTCR register is locked. This bit is cleared by hardware after detecting the unlock sequence.

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

3.9.10 Flash option control register (FLASH_OPTCR) for STM32F42xxx and STM32F43xxx

The FLASH_OPTCR register is used to modify the user option bytes.

Address offset: 0x14

Reset value: 0x0FFF AAED. The option bits are loaded with values from Flash memory at reset release.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPRMOD	DB1M	Reserved		nWRP[11:0]											
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
RDP[7:0]								nRST_STDBY	nRST_STOP	WDG_SW	BFB2	BOR_LEV		OPTSTRT	OPTLOCK
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rs	rs

Bit 31 **SPRMOD**: Selection of protection mode for nWPRi bits

0: PCROP disabled. nWPRi bits used for Write protection on sector i.

1: PCROP enabled. nWPRi bits used for PCROP protection on sector i

Bit 30 **DB1M**: Dual-bank on 1 Mbyte Flash memory devices

0: 1 Mbyte single bank Flash memory (contiguous addresses in bank1)

1: 1 Mbyte dual bank Flash memory. The Flash memory is organized as two banks of 512 Kbytes each (see *Table 7: 1 Mbyte Flash memory single bank vs dual bank organization (STM32F42xxx and STM32F43xxx)* and *Table 9: 1 Mbyte dual bank Flash memory organization (STM32F42xxx and STM32F43xxx)*). To perform an erase operation, the right sector must be programmed (see *Table 7* for information on the sector numbering scheme).

Note: If DB1M is set and an erase operation is performed on Bank 2 while the default sector number is selected (as an example, sector 8 is configured instead of sector 12), the erase operation on Bank 2 sector is not performed.

Bits 29:28 Reserved, must be kept cleared.

Bits 27:16 **nWRP[11:0]:** Not write protect

These bits contain the value of the write-protection and read-protection (PCROP) option bytes for sectors 0 to 11 after reset. They can be written to program a new write-protect or PCROP value into Flash memory.

If SPRMOD is reset:

0: Write protection active on sector i

1: Write protection not active on sector i

If SPRMOD is set:

0: PCROP protection not active on sector i

1: PCROP protection active on sector i

Bits 15:8 **RDP[7:0]:** Read protect

These bits contain the value of the read-protection option level after reset. They can be written to program a new read protection value into Flash memory.

0xAA: Level 0, read protection not active

0xCC: Level 2, chip read protection active

Others: Level 1, read protection of memories active

Bits 7:5 **USER:** User option bytes

These bits contain the value of the user option byte after reset. They can be written to program a new user option byte value into Flash memory.

Bit 7: nRST_STDBY
Bit 6: nRST_STOP
Bit 5: WDG_SW

Note: When changing the WDG mode from hardware to software or from software to hardware, a system reset is required to make the change effective.

Bit 4 **BFB2**: Dual-bank Boot option byte

0: Dual-bank boot disabled. Boot can be performed either from Flash memory bank 1 or from system memory depending on boot pin state (default)
1: Dual-bank boot enabled. Boot is always performed from system memory.

Note: For STM32F42xx and STM32F43xx 1MB part numbers, this option bit must be kept cleared when DB1M=0.

Bits 3:2 **BOR_LEV:** BOR reset Level

These bits contain the supply level threshold that activates/releases the reset. They can be written to program a new BOR level. By default, BOR is off. When the supply voltage (V_{DD}) drops below the selected BOR level, a device reset is generated.

00: BOR Level 3 (VBOR3), brownout threshold level 3
01: BOR Level 2 (VBOR2), brownout threshold level 2
10: BOR Level 1 (VBOR1), brownout threshold level 1
11: BOR off, POR/PDR reset threshold level is applied

Note: For full details on BOR characteristics, refer to the "Electrical characteristics" section of the product datasheet.

Bit 1 **OPTSTRT:** Option start

This bit triggers a user option operation when set. It is set only by software and cleared when the BSY bit is cleared.

Bit 0 **OPTLOCK:** Option lock

Write to 1 only. When this bit is set, it indicates that the FLASH_OPTCR register is locked. This bit is cleared by hardware after detecting the unlock sequence.

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

3.9.11 Flash option control register (FLASH_OPTCR1) for STM32F42xxx and STM32F43xxx

This register is available only on STM32F42xxx and STM32F43xxx.

The FLASH_OPTCR1 register is used to modify the user option bytes for bank 2.

Address offset: 0x18

Reset value: 0x0FFF 0000. The option bits are loaded with values from Flash memory at reset release.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				nWRP[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
Reserved															

Bits 31:28 Reserved, must be kept cleared.

Bits 27:16 **nWRP[11:0]:** Not write protect

These bits contain the value of the write-protection and read-protection (PCROP) option bytes for sectors 0 to 11 after reset. They can be written to program a new write-protect or PCROP value into Flash memory.

If SPRMOD is reset (default value):

0: Write protection active on sector i.

1: Write protection not active on sector i.

If SPRMOD is set:

0: PCROP protection not active on sector i.

1: PCROP protection active on sector i.

Bits 15:0 Reserved, must be kept cleared.

3.9.12 Flash interface register map

Table 19. Flash register map and reset values (STM32F405xx/07xx and STM32F415xx/17xx)

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**	Reserved																			DCRST	ICRST	DCEN	ICEN	PRFTEN	Reserved				LATENCY [2:0]		
	Reset value																				0	0	0	0	0					0	0	0
0x04	**FLASH_ KEYR**	KEY[31:16]															KEY[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
0x08	**FLASH_OPT KEYR**	OPTKEYR[31:16]															OPTKEYR[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
0x0C	**FLASH_SR**	Reserved															BSY	Reserved								PGSERR	PGPERR	PGAERR	WRPERR	Reserved		OPERR	EOP
	Reset value																0									0	0	0	0			0	0
0x10	**FLASH_CR**	LOCK	Reserved														EOPIE	Reserved							STRT	Reserved				PSIZE[1:0]		Reserved	SNB[3:0]				MER	SER	PG
	Reset value	1															0								0					0	0		0	0	0	0		0	0
0x14	**FLASH_ OPTCR**	Reserved															nWRP[11:0]									RDP[7:0]							nRST_STDBY	nRST_STOP	WDG_SW	Reserved	BOR_LEV[1:0]		OPTSTRT	OPTLOCK
	Reset value																1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	0	1	1	1		1	1	0	1

Table 20. Flash register map and reset values (STM32F42xxx and STM32F43xxx)

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**	Reserved																			DCRST	ICRST	DCEN	ICEN	PRFTEN	Reserved				LATENCY[3:0]			
	Reset value																				0	0	0	0	0					0	0	0	0
0x04	**FLASH_KEYR**	KEY[31:16]																KEY[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
0x08	**FLASH_ OPTKEYR**	OPTKEYR[31:16]																OPTKEYR[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
0x0C	**FLASH_SR**	Reserved															BSY	Reserved							RDERR	PGSERR	PGPERR	PGAERR	WRPERR	Reserved		OPERR	EOP
	Reset value																0								0	0	0	0	0			0	0

Table 20. Flash register map and reset values (STM32F42xxx and STM32F43xxx) (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
0x10	**FLASH_CR**	LOCK	Reserved						EOPIE	Reserved							STRT	MER1	Reserved					PSIZE[1:0]		SNB[4:0]					MER	SER	PG
	Reset value	1							0								0	0						0	0	0	0	0	0	0	0	0	0
0x14	**FLASH_OPTCR**	SPRMOD	DB1M	Reserved					nWRP[11:0]									RDP[7:0]								nRST_STDBY	nRST_STOP	WDG_SW	BFB2	BOR_LEV[1:0]		OPTSTRT	OPTLOCK
	Reset value	0	0			1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	0	1	1	1	0	1	1	0	1
0x18	**FLASH_ OPTCR1**	Reserved				nWRP[11:0]												Reserved															
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1																

4 CRC calculation unit

This section applies to the whole STM32F4xx family, unless otherwise specified.

4.1 CRC introduction

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

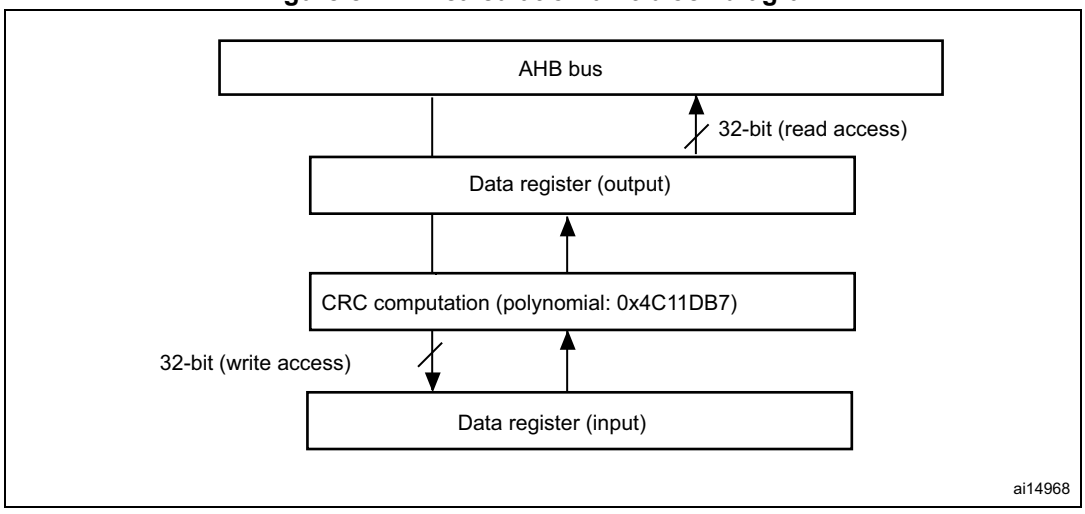
Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the EN/IEC 60335-1 standard, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared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

4.2 CRC main features

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

The block diagram is shown in *Figure 8*.

Figure 8. CRC calculation unit block diagram



4.3 CRC functional description

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

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

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

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

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

4.4 CRC registers

The CRC calculation unit contains two data registers and a control register. The peripheral
The CRC registers have to be accessed by words (32 bits).

4.4.1 Data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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

Bits 31:0 **Data register bits**

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

4.4.2 Independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

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

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

4.4.3 Control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

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

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

Bit 0 **RESET bit**

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

4.4.4 CRC register map

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

Table 21. CRC calculation unit register map and reset values

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

5 Power controller (PWR)

This section applies to the whole STM32F4xx family, unless otherwise specified.

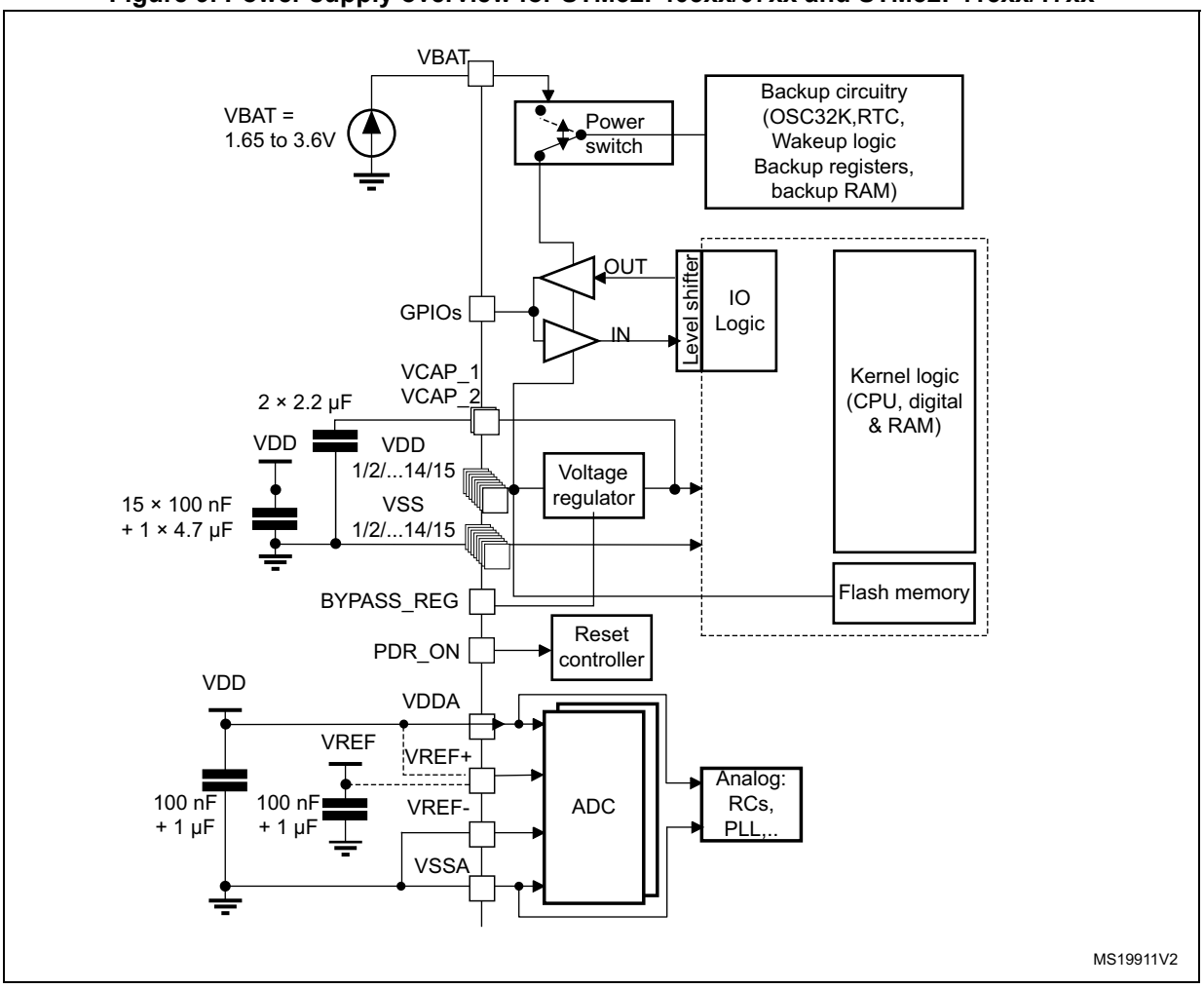
5.1 Power supplies

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

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

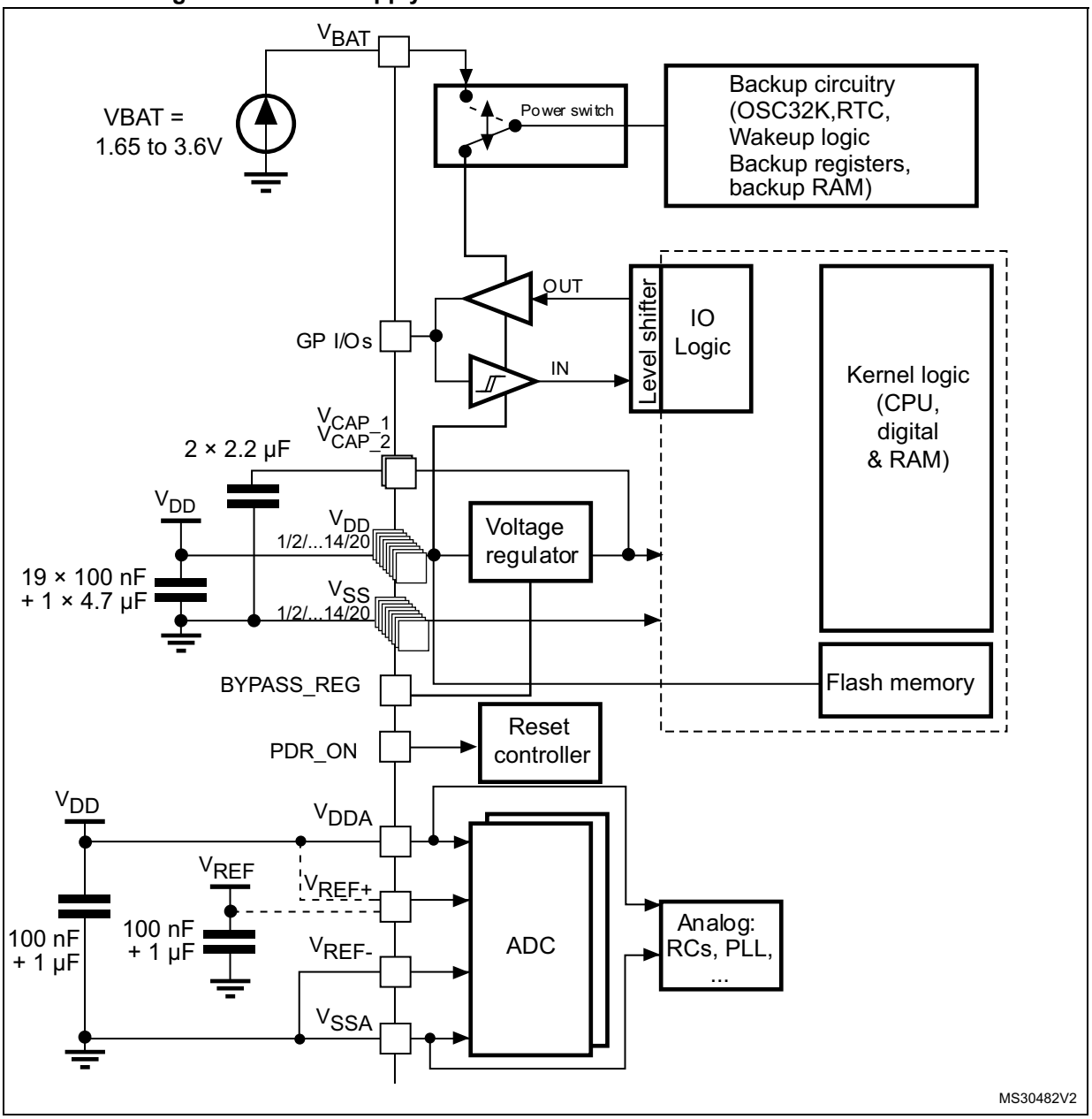
Note: Depending on the operating power supply range, some peripheral may be used with limited functionality and performance. For more details refer to section "General operating conditions" in STM32F4xx datasheets.

Figure 9. Power supply overview for STM32F405xx/07xx and STM32F415xx/17xx



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

Figure 10. Power supply overview for STM32F42xxx and STM32F43xxx



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

5.1.1 Independent A/D converter supply and reference voltage

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

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

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

5.1.2 Battery backup domain

Backup domain description

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

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

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

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

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

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

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

- PC14 and PC15 can be used as either GPIO or LSE pins
- PC13 can be used as a GPIOas the RTC_AF1 pin (refer to *Table 37: RTC_AF1 pin* for more details about this pin configuration)

Note: Due to the fact that the switch only sinks a limited amount of current (3 mA), the use of PI8 and PC13 to PC15 GPIOs in output mode is restricted: the speed has to be limited to 2 MHz with a maximum load of 30 pF and these I/Os must not be used as a current source (e.g. to drive an LED).

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

- PC14 and PC15 can be used as LSE pins only
- PC13 can be used as the RTC_AF1 pin (refer to *Table 37: RTC_AF1 pin* for more details about this pin configuration)
- PI8 can be used as RTC_AF2

Backup domain access

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

- Access to the RTC and RTC backup registers

1. Enable the power interface clock by setting the PWREN bits in the RCC_APB1ENR register (see *Section 7.3.13* and *Section 6.3.13*)
2. Set the DBP bit in the *Section 5.4.1* and *PWR power control register (PWR_CR) for STM32F42xxx and STM32F43xxx* to enable access to the backup domain
3. Select the RTC clock source: see *Section 7.2.8: RTC/AWU clock*
4. Enable the RTC clock by programming the RTCEN [15] bit in the *Section 7.3.20: RCC Backup domain control register (RCC_BDCR)*

- Access to the backup SRAM

1. Enable the power interface clock by setting the PWREN bits in the RCC_APB1ENR register (see *Section 7.3.13* and *Section 6.3.13* for STM32F405xx/07xx and STM32F415xx/17xx and STM32F42xxx and STM32F43xxx, respectively)
2. Set the DBP bit in the *PWR power control register (PWR_CR) for STM32F405xx/07xx and STM32F415xx/17xx* and *PWR power control register (PWR_CR) for STM32F42xxx and STM32F43xxx* to enable access to the backup domain
3. Enable the backup SRAM clock by setting BKPSRAMEN bit in the *RCC AHB1 peripheral clock enable register (RCC_AHB1ENR)*.

RTC and RTC backup registers

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar, two programmable alarm interrupts, and a periodic programmable wakeup flag with interrupt capability. The RTC contains 20 backup data registers (80 bytes) which are reset when a tamper detection event occurs. For more details refer to *Section 26: Real-time clock (RTC)*.

Backup SRAM

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

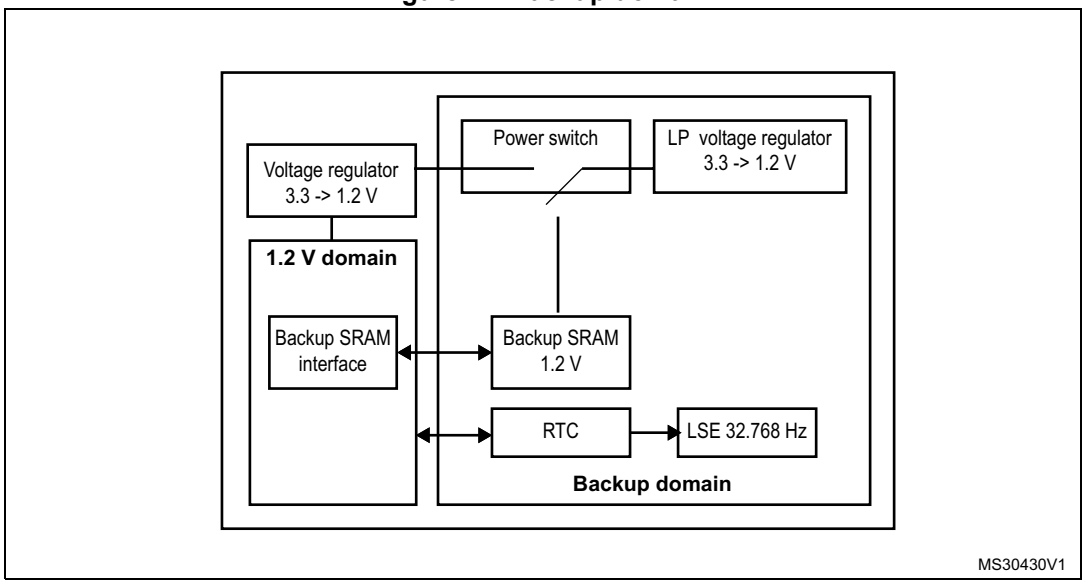
When the backup domain is supplied by V_{DD} (analog switch connected to V_{DD}), the backup SRAM is powered from V_{DD} which replaces the V_{BAT} power supply to save battery life.

When the backup domain is supplied by V_{BAT} (analog switch connected to V_{BAT} because V_{DD} is not present), the backup SRAM is powered by a dedicated low-power regulator. This regulator can be ON or OFF depending whether the application needs the backup SRAM function in Standby and V_{BAT} modes or not. The power-down of this regulator is controlled

by a dedicated bit, the BRE control bit of the PWR_CSR register (see *Section 5.4.2: PWR power control/status register (PWR_CSR) for STM32F405xx/07xx and STM32F415xx/17xx*).

The backup SRAM is not mass erased by an tamper event. When the Flash memory is read protected, the backup SRAM is also read protected to prevent confidential data, such as cryptographic private key, from being accessed. When the protection level change from level 1 to level 0 is requested, the backup SRAM content is erased.

Figure 11. Backup domain



5.1.3 Voltage regulator for STM32F405xx/07xx and STM32F415xx/17xx

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

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

When activated by software, the voltage regulator is always enabled after Reset. It works in three different modes depending on the application modes.

- In **Run mode**, the regulator supplies full power to the 1.2 V domain (core, memories and digital peripherals). In this mode, the regulator output voltage (around 1.2 V) can be scaled by software to different voltage values:

  Scale 1 or scale 2 can be configured on the fly through VOS (bit 15 of the PWR_CR register).

  The voltage scaling allows optimizing the power consumption when the device is clocked below the maximum system frequency.
- In **Stop mode,** the main regulator or the low-power regulator supplies to the 1.2 V domain, thus preserving the content of registers and internal SRAM. The voltage

regulator can be put either in main regulator mode (MR) or in low-power mode (LPR). The programmed voltage scale remains the same during Stop mode:

The programmed voltage scale remains the same during Stop mode (see *Section 5.4.1: PWR power control register (PWR_CR) for STM32F405xx/07xx and STM32F415xx/17xx*).

- In **Standby mode**, the regulator is powered down. The content of the registers and SRAM are lost except for the Standby circuitry and the backup domain.

Note: For more details, refer to the voltage regulator section in the STM32F405xx/07xx and STM32F415xx/17xx datasheets.

5.1.4 Voltage regulator for STM32F42xxx and STM32F43xxx

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

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

When activated by software, the voltage regulator is always enabled after Reset. It works in three different modes depending on the application modes (Run, Stop, or Standby mode).

- In **Run mode**, the main regulator supplies full power to the 1.2 V domain (core, memories and digital peripherals). In this mode, the regulator output voltage (around 1.2 V) can be scaled by software to different voltage values (scale 1, scale 2, and scale 3 can be configured through VOS[1:0] bits of the PWR_CR register). The scale can be modified only when the PLL is OFF and the HSI or HSE clock source is selected as system clock source. The new value programmed is active only when the PLL is ON. When the PLL is OFF, the voltage scale 3 is automatically selected.

  The voltage scaling allows optimizing the power consumption when the device is clocked below the maximum system frequency. After exit from Stop mode, the voltage

scale 3 is automatically selected.(see *Section 5.4.1: PWR power control register (PWR_CR) for STM32F405xx/07xx and STM32F415xx/17xx*.

2 operating modes are available:

- **Normal mode**: The CPU and core logic operate at maximum frequency at a given voltage scaling (scale 1, scale 2 or scale 3)
- **Over-drive mode**: This mode allows the CPU and the core logic to operate at a higher frequency than the normal mode for the voltage scaling scale 1 and scale 2.

- In **Stop mode**: the main regulator or low-power regulator supplies a low-power voltage to the 1.2V domain, thus preserving the content of registers and internal SRAM.

  The voltage regulator can be put either in main regulator mode (MR) or in low-power mode (LPR). Both modes can be configured by software as follows:

  - **Normal mode**: the 1.2 V domain is preserved in nominal leakage mode. It is the default mode when the main regulator (MR) or the low-power regulator (LPR) is enabled.
  - **Under-drive mode**: the 1.2 V domain is preserved in reduced leakage mode. This mode is only available with the main regulator or in low-power regulator mode (see *Table 22*).
- In **Standby mode**: the regulator is powered down. The content of the registers and SRAM are lost except for the Standby circuitry and the backup domain.

Note: Over-drive and under-drive mode are not available when the regulator is bypassed.

For more details, refer to the voltage regulator section in the STM32F42xxx and STM32F43xxx datasheets.

Table 22. Voltage regulator configuration mode versus device operating mode(1)

Voltage regulator configuration	Run mode	Sleep mode	Stop mode	Standby mode
Normal mode	MR	MR	MR or LPR	-
Over-drive mode(2)	MR	MR	-	-
Under-drive mode	-	-	MR or LPR	-
Power-down mode	-	-	-	Yes

1. '-' means that the corresponding configuration is not available.
2. The over-drive mode is not available when V_{DD} = 1.8 to 2.1 V.

Entering Over-drive mode

It is recommended to enter Over-drive mode when the application is not running critical tasks and when the system clock source is either HSI or HSE. To optimize the configuration time, enable the Over-drive mode during the PLL lock phase.

To enter Over-drive mode, follow the sequence below:

1. Select HSI or HSE as system clock.
2. Configure RCC_PLLCFGR register and set PLLON bit of RCC_CR register.
3. Set ODEN bit of PWR_CR register to enable the Over-drive mode and wait for the ODRDY flag to be set in the PWR_CSR register.
4. Set the ODSW bit in the PWR_CR register to switch the voltage regulator from Normal mode to Over-drive mode. The System will be stalled during the switch but the PLL clock system will be still running during locking phase.
5. Wait for the ODSWRDY flag in the PWR_CSR to be set.
6. Select the required Flash latency as well as AHB and APB prescalers.
7. Wait for PLL lock.
8. Switch the system clock to the PLL.
9. Enable the peripherals that are not generated by the System PLL (I2S clock, LCD-TFT clock, SAI1 clock, USB_48MHz clock....).

Note: The PLLI2S and PLLSAI can be configured at the same time as the system PLL.

During the Over-drive switch activation, no peripheral clocks should be enabled. The peripheral clocks must be enabled once the Over-drive mode is activated.

Entering Stop mode disables the Over-drive mode, as well as the PLL. The application software has to configure again the Over-drive mode and the PLL after exiting from Stop mode.

Exiting from Over-drive mode

It is recommended to exit from Over-drive mode when the application is not running critical tasks and when the system clock source is either HSI or HSE.There are two sequences that allow exiting from over-drive mode:

- By resetting simultaneously the ODEN and ODSW bits bit in the PWR_CR register (sequence 1)
- By resetting first the ODSW bit to switch the voltage regulator to Normal mode and then resetting the ODEN bit to disable the Over-drive mode (sequence 2).

Example of sequence 1:

1. Select HSI or HSE as system clock source.
2. Disable the peripheral clocks that are not generated by the System PLL (I2S clock, LCD-TFT clock, SAI1 clock, USB_48MHz clock,....)
3. Reset simultaneously the ODEN and the ODSW bits in the PWR_CR register to switch back the voltage regulator to Normal mode and disable the Over-drive mode.
4. Wait for the ODWRDY flag of PWR_CSR to be reset.

Example of sequence 2:

1. Select HSI or HSE as system clock source.
2. Disable the peripheral clocks that are not generated by the System PLL (I2S clock, LCD-TFT clock, SAI1 clock, USB_48MHz clock,....).
3. Reset the ODSW bit in the PWR_CR register to switch back the voltage regulator to Normal mode. The system clock is stalled during voltage switching.
4. Wait for the ODWRDY flag of PWR_CSR to be reset.
5. Reset the ODEN bit in the PWR_CR register to disable the Over-drive mode.

Note: During step 3, the ODEN bit remains set and the Over-drive mode is still enabled but not active (ODSW bit is reset). If the ODEN bit is reset instead, the Over-drive mode is disabled and the voltage regulator is switched back to the initial voltage.

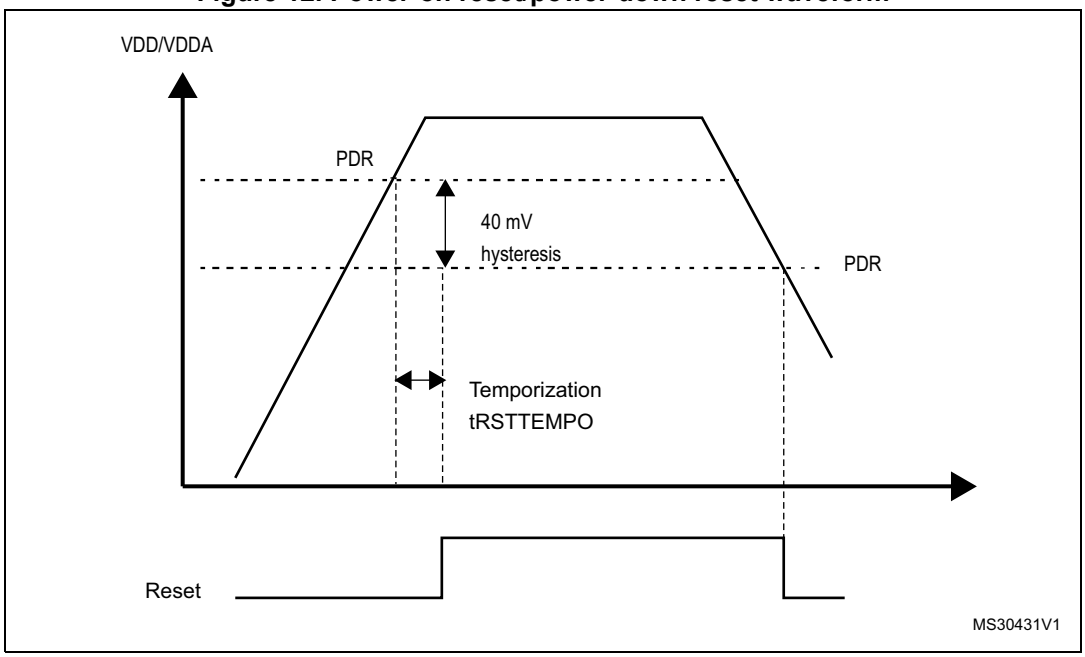
5.2 Power supply supervisor

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

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

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

Figure 12. Power-on reset/power-down reset waveform



5.2.2 Brownout reset (BOR)

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

V_{BOR} is configured through device option bytes. By default, BOR is off. 3 programmable V_{BOR} threshold levels can be selected:

- BOR Level 3 (VBOR3). Brownout threshold level 3.
- BOR Level 2 (VBOR2). Brownout threshold level 2.
- BOR Level 1 (VBOR1). Brownout threshold level 1.

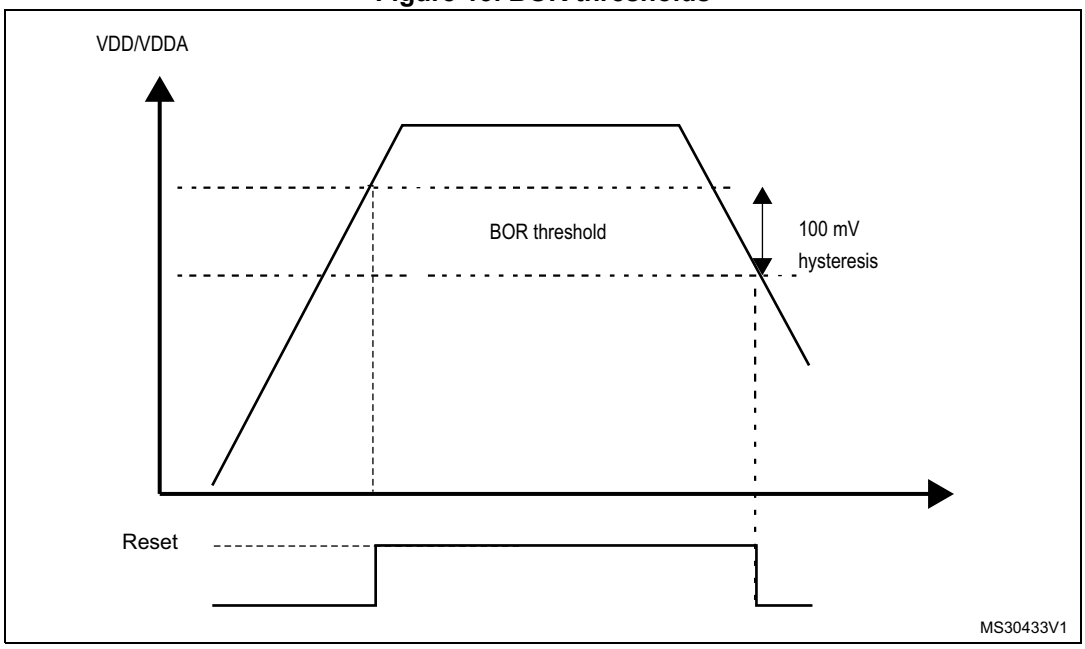
Note: For full details about BOR characteristics, refer to the "Electrical characteristics" section in the device datasheet.

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

The BOR can be disabled by programming the device option bytes. In this case, the power-on and power-down is then monitored by the POR/ PDR (see *Section 5.2.1: Power-on reset (POR)/power-down reset (PDR)*).

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

Figure 13. BOR thresholds



5.2.3 Programmable voltage detector (PVD)

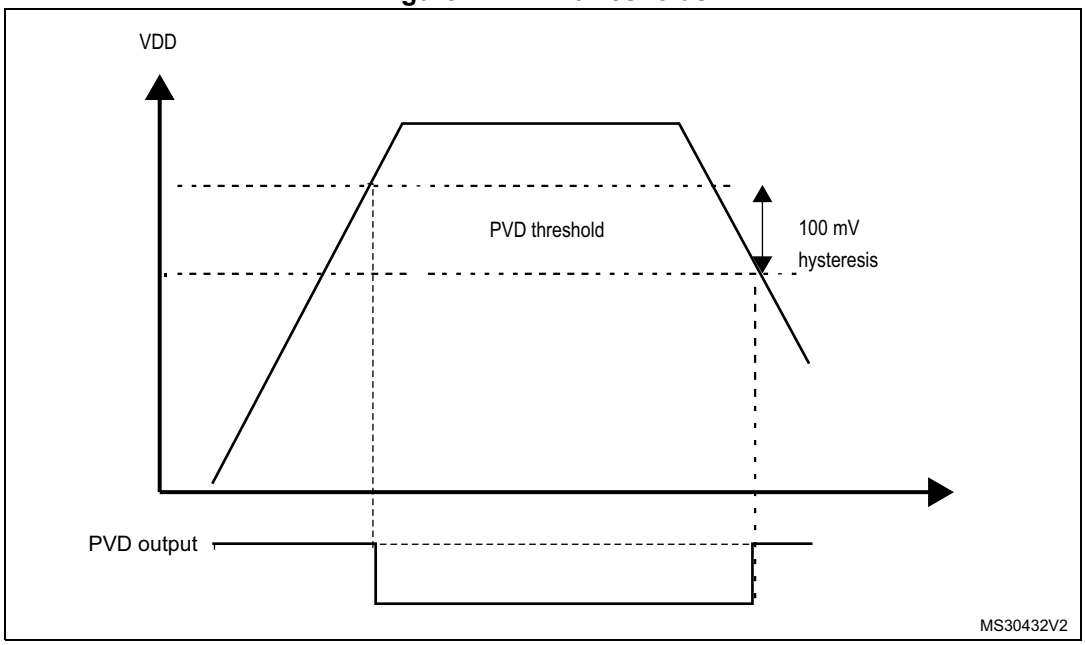
You can use the PVD to monitor the V_{DD} power supply by comparing it to a threshold selected by the PLS[2:0] bits in the *PWR power control register (PWR_CR) for*

STM32F405xx/07xx and STM32F415xx/17xx and *PWR power control register (PWR_CR) for STM32F42xxx and STM32F43xxx*.

The PVD is enabled by setting the PVDE bit.

A PVDO flag is available, in the *PWR power control/status register (PWR_CSR) for STM32F405xx/07xx and STM32F415xx/17xx*, to indicate if V_{DD} is higher or lower than the PVD threshold. This event is internally connected to the EXTI line16 and can generate an interrupt if enabled through the EXTI registers. The PVD output interrupt can be generated when V_{DD} drops below the PVD threshold and/or when V_{DD} rises above the PVD threshold depending on EXTI line16 rising/falling edge configuration. As an example the service routine could perform emergency shutdown tasks.

Figure 14. PVD thresholds



5.3 Low-power modes

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

The devices feature three low-power modes:

- Sleep mode (Cortex®-M4 with FPU core stopped, peripherals kept running)
- Stop mode (all clocks are stopped)
- Standby mode (1.2 V domain powered off)

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

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

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®-M4 with FPU 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

The MCU exits from Sleep and Stop modes 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®-M4 with FPU 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®-M4 with FPU 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

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

The MCU exits from Standby 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 237: RTC block diagram*).

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

Only enabled NVIC interrupts with sufficient priority will wakeup and interrupt the MCU.

Table 23. Low-power mode summary

Mode name	Entry	Wakeup	Effect on 1.2 V domain clocks	Effect on V_{DD} domain clocks	Voltage regulator
Sleep (Sleep now or Sleep-on-exit)	WFI or Return from ISR	Any interrupt	CPU CLK OFF no effect on other clocks or analog clock sources	None	ON
	WFE	Wakeup event			
Stop	PDDS and LPDS bits + SLEEPDEEP bit + WFI, Return from ISR or WFE	Any EXTI line (configured in the EXTI registers, internal and external lines)	All 1.2 V domain clocks OFF	HSI and HSE oscillators OFF	ON or in low- power mode (depends on *PWR power control register (PWR_CR) for STM32F405xx/07xx and STM32F415xx/17xx* and *PWR power control register (PWR_CR) for STM32F405xx/07xx and STM32F415xx/17xxPWR power control register (PWR_CR) for STM32F42xxx and STM32F43xxx*
Standby	PDDS bit + SLEEPDEEP bit + WFI, Return from ISR or WFE	WKUP pin rising edge, RTC alarm (Alarm A or Alarm B), RTC Wakeup event, RTC tamper events, RTC time stamp event, external reset in NRST pin, IWDG reset			OFF

5.3.1 Slowing down system clocks

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

For more details refer to *Section 7.3.3: RCC clock configuration register (RCC_CFGR)*.

5.3.2 Peripheral clock gating

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

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

Peripheral clock gating is controlled by the AHB1 peripheral clock enable register (RCC_AHB1ENR), AHB2 peripheral clock enable register (RCC_AHB2ENR), AHB3 peripheral clock enable register (RCC_AHB3ENR) (see *Section 7.3.10: RCC AHB1 peripheral clock enable register (RCC_AHB1ENR)*, *Section 7.3.11: RCC AHB2 peripheral clock enable register (RCC_AHB2ENR)*, *Section 7.3.12: RCC AHB3 peripheral clock enable register (RCC_AHB3ENR)* for STM32F405xx/07xx and STM32F415xx/17xx, and *Section 6.3.10: RCC AHB1 peripheral clock register (RCC_AHB1ENR)*, *Section 6.3.11: RCC AHB2 peripheral clock enable register (RCC_AHB2ENR)*, and *Section 6.3.12: RCC AHB3 peripheral clock enable register (RCC_AHB3ENR)* for STM32F42xxx and STM32F43xxx).

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

5.3.3 Sleep mode

Entering Sleep mode

The Sleep mode is entered according to *Section : Entering low-power mode*, when the SLEEPDEEP bit in the Cortex®-M4 with FPU System Control register is cleared.

Refer to *Table 24* and *Table 25* for details on how to enter Sleep mode.

Note: All interrupt pending bits must be cleared before the sleep mode entry.

Exiting Sleep mode

The Sleep mode is exited according to *Section : Exiting low-power mode*.

Refer to *Table 24* and *Table 25* for more details on how to exit Sleep mode.

Table 24. Sleep-now entry and exit

Sleep-now mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0, and – No interrupt (for WFI) or event (for WFE) is pending. Refer to the Cortex®-M4 with FPU System Control register.
	On Return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1, – No interrupt is pending. Refer to the Cortex®-M4 with FPU System Control register.

Table 24. Sleep-now entry and exit (continued)

Sleep-now mode	Description
Mode exit	If WFI or Return from ISR was used for entry: Interrupt: Refer to *Table 61: Vector table for STM32F405xx/07xx and STM32F415xx/17xx* and *Table 62: Vector table for STM32F42xxx and STM32F43xxx* If WFE was used for entry and SEVONPEND = 0 Wakeup event: Refer to *Section 12.2.3: Wakeup event management* f WFE was used for entry and SEVONPEND = 1 Interrupt even when disabled in NVIC: refer to *Table 61: Vector table for STM32F405xx/07xx and STM32F415xx/17xx* and *Table 62: Vector table for STM32F42xxx and STM32F43xxx* or Wakeup event (see *Section 12.2.3: Wakeup event management*).
Wakeup latency	None

Table 25. Sleep-on-exit entry and exit

Sleep-on-exit	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0, and – No interrupt (for WFI) or event (for WFE) is pending. Refer to the Cortex®-M4 with FPU System Control register.
	On Return from ISR while: – SLEEPDEEP = 0, and – SLEEPONEXIT = 1, and – No interrupt is pending. Refer to the Cortex®-M4 with FPU System Control register.
Mode exit	Interrupt: refer to *Table 61: Vector table for STM32F405xx/07xx and STM32F415xx/17xx* and *Table 62: Vector table for STM32F42xxx and STM32F43xxx*
Wakeup latency	None

5.3.4 Stop mode (STM32F405xx/07xx and STM32F415xx/17xx)

The Stop mode is based on the Cortex®-M4 with FPU deepsleep mode combined with peripheral clock gating. The voltage regulator can be configured either in normal or low-power mode. In Stop mode, all clocks in the 1.2 V domain are stopped, the PLLs, the HSI and the HSE RC oscillators are disabled. Internal SRAM and register contents are preserved.

By setting the FPDS bit in the PWR_CR register, the Flash memory also enters power-down mode when the device enters Stop mode. When the Flash memory is in power-down mode, an additional startup delay is incurred when waking up from Stop mode (see *Table 26: Stop operating modes (STM32F405xx/07xx and STM32F415xx/17xx)* and *Section 5.4.1: PWR power control register (PWR_CR) for STM32F405xx/07xx and STM32F415xx/17xx*).

Table 26. Stop operating modes (STM32F405xx/07xx and STM32F415xx/17xx)

Stop mode	LPDS bit	FPDS bit	Wake-up latency
STOP MR (Main regulator)	0	0	HSI RC startup time
STOP MR-FPD	0	1	HSI RC startup time + Flash wakeup time from Power Down mode
STOP LP	1	0	HSI RC startup time + regulator wakeup time from LP mode
STOP LP-FPD	1	1	HSI RC startup time + Flash wakeup time from Power Down mode + regulator wakeup time from LP mode

Entering Stop mode (for STM32F405xx/07xx and STM32F415xx/17xx)

The Stop mode is entered according to *Section : Entering low-power mode*, when the SLEEPDEEP bit in the Cortex®-M4 with FPU System Control register is set.

Refer to *Table 27* for details on how to enter the Stop mode.

To further reduce power consumption in Stop mode, the internal voltage regulator can be put in low-power mode. This is configured by the LPDS bit of the *PWR power control register (PWR_CR) for STM32F405xx/07xx and STM32F415xx/17xx* and *PWR power control register (PWR_CR) for STM32F42xxx and STM32F43xxx*.

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

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

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

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a Reset. See *Section 21.3* in *Section 21: Independent watchdog (IWDG)*.
- Real-time clock (RTC): this is configured by the RTCEN bit in the *Section 7.3.20: RCC Backup domain control register (RCC_BDCR)*
- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the *Section 7.3.21: RCC clock control & status register (RCC_CSR)*.
- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the *Section 7.3.20: RCC Backup domain control register (RCC_BDCR)*.

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

Note: If the application needs to disable the external clock before entering Stop mode, the HSEON bit must first be disabled and the system clock switched to HSI.

Otherwise, if the HSEON bit is kept enabled while the external clock (external oscillator) can be removed before entering stop mode, the clock security system (CSS) feature must be enabled to detect any external oscillator failure and avoid a malfunction behavior when entering stop mode.

Exiting Stop mode (for STM32F405xx/07xx and STM32F415xx/17xx)

The Stop mode is exited according to *Section : Exiting low-power mode*.

Refer to *Table 27* for more details on how to exit Stop mode.

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

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

Table 27. Stop mode entry and exit (for STM32F405xx/07xx and STM32F415xx/17xx)

Stop mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – No interrupt (for WFI) or event (for WFE) is pending, – SLEEPDEEP bit is set in Cortex®-M4 with FPU System Control register, – PDDS bit is cleared in Power Control register (PWR_CR), – Select the voltage regulator mode by configuring LPDS bit in PWR_CR.
	On Return from ISR: – No interrupt is pending, – SLEEPDEEP bit is set in Cortex®-M4 with FPU System Control register, – SLEEPONEXIT = 1, – PDDS bit is cleared in Power Control register (PWR_CR).
	Note: To enter Stop mode, all EXTI Line pending bits (in Pending register (EXTI_PR)), all peripheral interrupts pending bits, the RTC Alarm (Alarm A and Alarm B), RTC wakeup, RTC tamper, and RTC time stamp flags, must be reset. Otherwise, the Stop mode entry procedure is ignored and program execution continues.

Table 27. Stop mode entry and exit (for STM32F405xx/07xx and STM32F415xx/17xx)

Stop mode	Description
Mode exit	If WFI or Return from ISR was used for entry: Any EXTI lines 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 *Table 61: Vector table for STM32F405xx/07xx and STM32F415xx/17xx on page 372* and *Table 62: Vector table for STM32F42xxx and STM32F43xxx*. If WFE was used for entry and SEVONPEND = 0 Any EXTI lines configured in event mode. Refer to *Section 12.2.3: Wakeup event management on page 380*. If WFE was used for entry and SEVONPEND = 1: – Any EXTI lines configured in Interrupt mode (even if the corresponding EXTI Interrupt vector is disabled in the NVIC). The interrupt source can be an external interrupt or a peripheral with wakeup capability. Refer to *Table 61: Vector table for STM32F405xx/07xx and STM32F415xx/17xx on page 372* and *Table 62: Vector table for STM32F42xxx and STM32F43xxx*. – Wakeup event: refer to *Section 12.2.3: Wakeup event management on page 380*.
Wakeup latency	*Table 26: Stop operating modes (STM32F405xx/07xx and STM32F415xx/17xx)*

5.3.5 Stop mode (STM32F42xxx and STM32F43xxx)

The Stop mode is based on the Cortex®-M4 with FPU deepsleep mode combined with peripheral clock gating. The voltage regulator can be configured either in normal or low-power mode. In Stop mode, all clocks in the 1.2 V domain are stopped, the PLLs, the HSI and the HSE RC oscillators are disabled. Internal SRAM and register contents are preserved.

In Stop mode, the power consumption can be further reduced by using additional settings in the PWR_CR register. However this will induce an additional startup delay when waking up from Stop mode (see *Table 28*).

Table 28. Stop operating modes (STM32F42xxx and STM32F43xxx)

Voltage Regulator Mode		UDEN[1:0] bits	MRUDS bit	LPUDS bit	LPDS bit	FPDS bit	Wakeup latency
Normal mode	STOP MR (Main Regulator)	-	0	-	0	0	HSI RC startup time
	STOP MR- FPD	-	0	-	0	1	HSI RC startup time + Flash wakeup time from power-down mode
	STOP LP	-	0	0	1	0	HSI RC startup time + regulator wakeup time from LP mode
	STOP LP-FPD	-	-	0	1	1	HSI RC startup time + Flash wakeup time from power-down mode + regulator wakeup time from LP mode
Under-drive Mode	STOP UMR-FPD	3	1	-	0	-	HSI RC startup time + Flash wakeup time from power-down mode + Main regulator wakeup time from under-drive mode + Core logic to nominal mode
	STOP ULP-FPD	3	-	1	1	-	HSI RC startup time + Flash wakeup time from power-down mode + regulator wakeup time from LP under-drive mode + Core logic to nominal mode

Entering Stop mode (STM32F42xxx and STM32F43xxx)

The Stop mode is entered according to *Section : Entering low-power mode*, when the SLEEPDEEP bit in the Cortex®-M4 with FPU System Control register is set.

Refer to *Table 29* for details on how to enter the Stop mode.

When the microcontroller enters in Stop mode, the voltage scale 3 is automatically selected. To further reduce power consumption in Stop mode, the internal voltage regulator can be put in low voltage mode. This is configured by the LPDS, MRUDS, LPUDS and UDEN bits of the *PWR power control register (PWR_CR) for STM32F405xx/07xx and STM32F415xx/17xx*.

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

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

If the Over-drive mode was enabled before entering Stop mode, it is automatically disabled during when the Stop mode is activated.

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

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a Reset. See *Section 21.3* in *Section 21: Independent watchdog (IWDG)*.
- Real-time clock (RTC): this is configured by the RTCEN bit in the *Section 7.3.20: RCC Backup domain control register (RCC_BDCR)*
- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the *Section 7.3.21: RCC clock control & status register (RCC_CSR)*.
- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the *RCC Backup domain control register (RCC_BDCR)*.

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

Note: Before entering Stop mode, it is recommended to enable the clock security system (CSS) feature to prevent external oscillator (HSE) failure from impacting the internal MCU behavior.

Exiting Stop mode (STM32F42xxx and STM32F43xxx)

The Stop mode is exited according to *Section : Exiting low-power mode*.

Refer to *Table 29* for more details on how to exit Stop mode.

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

If the Under-drive mode was enabled, it is automatically disabled after exiting Stop mode.

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

When the voltage regulator operates in Under-drive mode, an additional startup delay is induced when waking up from Stop mode.

Table 29. Stop mode entry and exit (STM32F42xxx and STM32F43xxx)

Stop mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – No interrupt or event is pending, – SLEEPDEEP bit is set in Cortex®-M4 with FPU System Control register, – PDDS bit is cleared in Power Control register (PWR_CR), – Select the voltage regulator mode by configuring LPDS, MRUDS, LPUDS and UDEN bits in PWR_CR (see *Table 28: Stop operating modes (STM32F42xxx and STM32F43xxx)*).
	On Return from ISR while: – No interrupt is pending, – SLEEPDEEP bit is set in Cortex®-M4 with FPU System Control register, and – SLEEPONEXIT = 1, and – PDDS is cleared in PWR_CR1.
	Note: To enter Stop mode, all EXTI Line pending bits (in Pending register (EXTI_PR)), all peripheral interrupts pending bits, the RTC Alarm (Alarm A and Alarm B), RTC wakeup, RTC tamper, and RTC time stamp flags, must be reset. Otherwise, the Stop mode entry procedure is ignored and program execution continues.
Mode exit	If WFI or Return from ISR was used for entry: All EXTI lines 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 *Table 61: Vector table for STM32F405xx/07xx and STM32F415xx/17xx on page 372*. If WFE was used for entry and SEVONPEND = 0: All EXTI Lines configured in event mode. Refer to *Section 12.2.3: Wakeup event management on page 380* If WFE was used for entry and SEVONPEND = 1: – Any EXTI lines 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 *Table 61: Vector table for STM32F405xx/07xx and STM32F415xx/17xx on page 372* and *Table 62: Vector table for STM32F42xxx and STM32F43xxx*. – Wakeup event: refer to *Section 12.2.3: Wakeup event management on page 380*.
Wakeup latency	Refer to *Table 28: Stop operating modes (STM32F42xxx and STM32F43xxx)*

5.3.6 Standby mode

The Standby mode allows to achieve the lowest power consumption. It is based on the Cortex®-M4 with FPU deepsleep mode, with the voltage regulator disabled. The 1.2 V domain is consequently powered off. The PLLs, the HSI oscillator and the HSE oscillator are also switched off. SRAM and register contents are lost except for registers in the backup domain (RTC registers, RTC backup register and backup SRAM), and Standby circuitry (see *Figure 9*).

Entering Standby mode

The Standby mode is entered according to *Section : Entering low-power mode*, when the SLEEPDEEP bit in the Cortex®-M4 with FPU System Control register is set.

Refer to *Table 30* for more details on how to enter Standby mode.

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

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a reset. See *Section 21.3* in *Section 21: Independent watchdog (IWDG)*.
- Real-time clock (RTC): this is configured by the RTCEN bit in the backup domain control register (RCC_BDCR)
- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the Control/status register (RCC_CSR).
- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the backup domain control register (RCC_BDCR)

Exiting Standby mode

The Standby mode is exited according to *Section : Exiting low-power mode*. The SBF status flag in PWR_CR (see *Section 5.4.2: PWR power control/status register (PWR_CSR) for STM32F405xx/07xx and STM32F415xx/17xx*) indicates that the MCU was in Standby mode. All registers are reset after wakeup from Standby except for PWR_CR.

Refer to *Table 30* for more details on how to exit Standby mode.

Table 30. Standby mode entry and exit

Standby mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP is set in Cortex®-M4 with FPU System Control register, – PDDS bit is set in Power Control register (PWR_CR), – No interrupt (for WFI) or event (for WFE) is pending, – WUF bit is cleared in Power Control register (PWR_CR), – the RTC flag corresponding to the chosen wakeup source (RTC Alarm A, RTC Alarm B, RTC wakeup, Tamper or Timestamp flags) is cleared
	On return from ISR while: – SLEEPDEEP bit is set in Cortex®-M4 with FPU System Control register, and – SLEEPONEXIT = 1, and – PDDS bit is set in Power Control register (PWR_CR), and – No interrupt is pending, – WUF bit is cleared in Power Control/Status register (PWR_SR), – The RTC flag corresponding to the chosen wakeup source (RTC Alarm A, RTC Alarm B, RTC wakeup, Tamper or Timestamp flags) is cleared.
Mode exit	WKUP pin rising edge, RTC alarm (Alarm A and Alarm B), RTC wakeup, tamper event, time stamp event, external reset in NRST pin, IWDG reset.
Wakeup latency	Reset phase.

I/O states in Standby mode

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

- Reset pad (still available)
- RTC_AF1 pin (PC13) if configured for tamper, time stamp, RTC Alarm out, or RTC clock calibration out
- WKUP pin (PA0), if enabled

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop or Standby mode while the debug features are used. This is due to the fact that the Cortex®-M4 with FPU 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 38.16.1: Debug support for low-power modes*.

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

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

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

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

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

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

For this purpose, two of the three alternate RTC clock sources can be selected by programming the RTCSEL[1:0] bits in the *Section 7.3.20: RCC Backup domain control register (RCC_BDCR)*:

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

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

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

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

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

Safe RTC alternate function wakeup flag clearing sequence

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

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

- When using RTC alarm to wake up the device from the low-power modes:
  a) Disable the RTC alarm interrupt (ALRAIE or ALRBIE bits in the RTC_CR register)
  b) Clear the RTC alarm (ALRAF/ALRBF) flag

c) Clear the PWR Wakeup (WUF) flag
d) Enable the RTC alarm interrupt
e) Re-enter the low-power mode

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

5.4 Power control registers (STM32F405xx/07xx and STM32F415xx/17xx)

5.4.1 PWR power control register (PWR_CR) for STM32F405xx/07xx and STM32F415xx/17xx

Address offset: 0x00

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

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

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

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

This bit controls the main internal voltage regulator output voltage to achieve a trade-off between performance and power consumption when the device does not operate at the maximum frequency.

0: Scale 2 mode

1: Scale 1 mode (default value at reset)

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

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

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

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

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

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

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

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

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

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

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

000: 2.0 V

001: 2.1 V

010: 2.3 V

011: 2.5 V

100: 2.6 V

101: 2.7 V

110: 2.8 V

111: 2.9 V

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

Bit 4 **PVDE:** Power voltage detector enable
This bit is set and cleared by software.
0: PVD disabled
1: PVD enabled

Bit 3 **CSBF**: Clear standby flag
This bit is always read as 0.
0: No effect
1: Clear the SBF Standby Flag (write).

Bit 2 **CWUF:** Clear wakeup flag
This bit is always read as 0.
0: No effect
1: Clear the WUF Wakeup Flag **after 2 System clock cycles**

Bit 1 **PDDS**: Power-down deepsleep
This bit is set and cleared by software. It works together with the LPDS bit.
0: Enter Stop mode when the CPU enters deepsleep. The regulator status depends on the LPDS bit.
1: Enter Standby mode when the CPU enters deepsleep.

Bit 0 **LPDS:** Low-power deepsleep
This bit is set and cleared by software. It works together with the PDDS bit.
0: Voltage regulator on during Stop mode
1: Voltage regulator in low-power mode during Stop mode

5.4.2 PWR power control/status register (PWR_CSR) for STM32F405xx/07xx and STM32F415xx/17xx

Address offset: 0x04

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

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

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

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

Bit 14 **VOSRDY**: Regulator voltage scaling output selection ready bit
0: Not ready
1: Ready

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

Bit 9 **BRE**: Backup regulator enable

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

0: Backup regulator disabled

1: Backup regulator enabled

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

Bit 8 **EWUP**: Enable WKUP pin

This bit is set and cleared by software.

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

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

Note: This bit is reset by a system reset.

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

Bit 3 **BRR**: Backup regulator ready

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

0: Backup Regulator not ready

1: Backup Regulator ready

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

Bit 2 **PVDO:** PVD output

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

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

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

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

Bit 1 **SBF:** Standby flag

This bit is set by hardware and cleared only by a POR/PDR (power-on reset/power-down reset) or by setting the CSBF bit in the PWR_CR register.

0: Device has not been in Standby mode

1: Device has been in Standby mode

Bit 0 **WUF:** Wakeup flag

This bit is set by hardware and cleared either by a system reset or by setting the CWUF bit in the PWR_CR register.

0: No wakeup event occurred

1: A wakeup event was received from the WKUP pin or from the RTC alarm (Alarm A or Alarm B), RTC Tamper event, RTC TimeStamp event or RTC Wakeup).

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

5.5 Power control registers (STM32F42xxx and STM32F43xxx)

5.5.1 PWR power control register (PWR_CR) for STM32F42xxx and STM32F43xxx

Address offset: 0x00

Reset value: 0x0000 C000 (reset by wakeup from Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												UDEN[1:0]		ODSWEN	ODEN
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VOS[1:0]		ADCDC1	Res.	MRUDS	LPUDS	FPDS	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rc_w1	rc_w1	rw	rw

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

Bits 19:18 **UDEN[1:0]**: Under-drive enable in stop mode

These bits are set by software. They allow to achieve a lower power consumption in Stop mode but with a longer wakeup time.

When set, the digital area has less leakage consumption when the device enters Stop mode.

00: Under-drive disable

01: Reserved

10: Reserved

11:Under-drive enable

Bit 17 **ODSWEN**: Over-drive switching enabled.

This bit is set by software. It is cleared automatically by hardware after exiting from Stop mode or when the ODEN bit is reset. When set, It is used to switch to Over-drive mode.

To set or reset the ODSWEN bit, the HSI or HSE must be selected as system clock.

The ODSWEN bit must only be set when the ODRDY flag is set to switch to Over-drive mode.

0: Over-drive switching disabled

1: Over-drive switching enabled

Note: On any over-drive switch (enabled or disabled), the system clock will be stalled during the internal voltage set up.

Bit 16 **ODEN**: Over-drive enable

This bit is set by software. It is cleared automatically by hardware after exiting from Stop mode. It is used to enabled the Over-drive mode in order to reach a higher frequency.

To set or reset the ODEN bit, the HSI or HSE must be selected as system clock. When the ODEN bit is set, the application must first wait for the Over-drive ready flag (ODRDY) to be set before setting the ODSWEN bit.

0: Over-drive disabled

1: Over-drive enabled

Bits 15:14 **VOS[1:0]**: Regulator voltage scaling output selection

These bits control the main internal voltage regulator output voltage to achieve a trade-off between performance and power consumption when the device does not operate at the maximum frequency (refer to the STM32F42xx and STM32F43xx datasheets for more details).

These bits can be modified only when the PLL is OFF. The new value programmed is active only when the PLL is ON. When the PLL is OFF, the voltage scale 3 is automatically selected.

00: Reserved (Scale 3 mode selected)

01: Scale 3 mode

10: Scale 2 mode

11: Scale 1 mode (reset value)

Bit 13 **ADCDC1**:

0: No effect.

1: Refer to AN4073 for details on how to use this bit.

Note: This bit can only be set when operating at supply voltage range 2.7 to 3.6V and when the Prefetch is OFF.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **MRUDS**: Main regulator in deepsleep under-drive mode

This bit is set and cleared by software.

0: Main regulator ON when the device is in Stop mode

1: Main Regulator in under-drive mode and Flash memory in power-down when the device is in Stop under-drive mode.

Bit 10 **LPUDS**: Low-power regulator in deepsleep under-drive mode

This bit is set and cleared by software.

0: Low-power regulator ON if LPDS bit is set when the device is in Stop mode

1: Low-power regulator in under-drive mode if LPDS bit is set and Flash memory in power-down when the device is in Stop under-drive mode.

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

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

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

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

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

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

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

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

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

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

000: 2.0 V
001: 2.1 V
010: 2.3 V
011: 2.5 V
100: 2.6 V
101: 2.7 V
110: 2.8 V
111: 2.9 V

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

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

This bit is set and cleared by software.

0: PVD disabled
1: PVD enabled

Bit 3 **CSBF**: Clear standby flag

This bit is always read as 0.

0: No effect
1: Clear the SBF Standby Flag (write).

Bit 2 **CWUF:** Clear wakeup flag

This bit is always read as 0.

0: No effect
1: Clear the WUF Wakeup Flag **after 2 System clock cycles**

Bit 1 **PDDS**: Power-down deepsleep

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

0: Enter Stop mode when the CPU enters deepsleep. The regulator status depends on the LPDS bit.
1: Enter Standby mode when the CPU enters deepsleep.

Bit 0 **LPDS:** Low-power deepsleep

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

0:Main voltage regulator ON during Stop mode
1: Low-power voltage regulator ON during Stop mode

5.5.2 PWR power control/status register (PWR_CSR) for STM32F42xxx and STM32F43xxx

Address offset: 0x04

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												UDRDY[1:0]		ODSWRDY	ODRDY
												rc_w1	rc_w1	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	VOS RDY	Reserved				BRE	EWUP	Reserved.				BRR	PVDO	SBF	WUF
	r					rw	rw					r	r	r	r

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

Bits 19:18 **UDRDY[1:0]**: Under-drive ready flag

These bits are set by hardware when MCU entered stop Under-drive mode and exited. When the under-drive mode is enabled, these bits are not set as long as the MCU has not entered stop mode yet. They are cleared by programming them to 1.

00: Under-drive is disabled

01: Reserved

10: Reserved

11:Under-drive mode is activated in Stop mode.

Bit 17 **ODSWRDY**: Over-drive mode switching ready

0: Over-drive mode is not active.

1: Over-drive mode is active on digital area on 1.2 V domain

Bit 16 **ODRDY**: Over-drive mode ready

0: Over-drive mode not ready.

1: Over-drive mode ready

Bit 14 **VOSRDY**: Regulator voltage scaling output selection ready bit

0: Not ready

1: Ready

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

Bit 9 **BRE**: Backup regulator enable

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

0: Backup regulator disabled

1: Backup regulator enabled

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

Bit 8 **EWUP**: Enable WKUP pin

This bit is set and cleared by software.

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

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

Note: This bit is reset by a system reset.

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

Bit 3 **BRR**: Backup regulator ready

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

0: Backup Regulator not ready

1: Backup Regulator ready

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

Bit 2 **PVDO:** PVD output

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

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

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

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

Bit 1 **SBF:** Standby flag

This bit is set by hardware and cleared only by a POR/PDR (power-on reset/power-down reset) or by setting the CSBF bit in the *PWR power control register (PWR_CR) for STM32F405xx/07xx and STM32F415xx/17xx*

0: Device has not been in Standby mode

1: Device has been in Standby mode

Bit 0 **WUF:** Wakeup flag

This bit is set by hardware and cleared either by a system reset or by setting the CWUF bit in the PWR_CR register.

0: No wakeup event occurred

1: A wakeup event was received from the WKUP pin or from the RTC alarm (Alarm A or Alarm B), RTC Tamper event, RTC TimeStamp event or RTC Wakeup).

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

5.6 PWR register map

The following table summarizes the PWR registers.

Table 31. PWR - register map and reset values for STM32F405xx/07xx and STM32F415xx/17xx

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**PWR_CR**	Reserved																	VOS	Reserved				FPDS	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	Reset value																		1					0	0	0	0	0	0	0	0	0	0
0x004	**PWR_CSR**	Reserved																	VOSRDY	Reserved				BRE	EWUP	Reserved				BRR	PVDO	SBF	WUF
	Reset value																		0					0	0					0	0	0	0

Table 32. PWR - register map and reset values for STM32F42xxx and STM32F43xxx

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PWR_CR	Reserved												UDEN[1:0]		ODSWEN	ODEN	VOS[1:0]		ADCDC1	Reserved	MRUDS	LPUDS	FPDS	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	Reset value													1	1	1	1	1	1	0		1	1	0	0	0	0	0	0	0	0	0	0
0x004	PWR_CSR	Reserved												UDRDY[1:0]		ODSWRDY	ODRDY	Reserved	VOSRDY	Reserved				BRE	EWUP	Reserved				BRR	PVDO	SBF	WUF
	Reset value													0	0	0	0		0					0	0					0	0	0	0

Refer to *Section 2.3: Memory map* for the register boundary addresses.

6 Reset and clock control for STM32F42xxx and STM32F43xxx (RCC)

6.1 Reset

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

6.1.1 System reset

A system reset sets all registers to their reset values except the reset flags in the clock controller CSR register and the registers in the Backup domain (see *Figure 15*).

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

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

Software reset

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

The SYSRESETREQ bit in Cortex®-M4 with FPU Application Interrupt and Reset Control Register must be set to force a software reset on the device. Refer to the Cortex®-M4 with FPU technical reference manual for more details.

Low-power management reset

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

1. Reset generated when entering the Standby mode:

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

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

6.1.2 Power reset

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

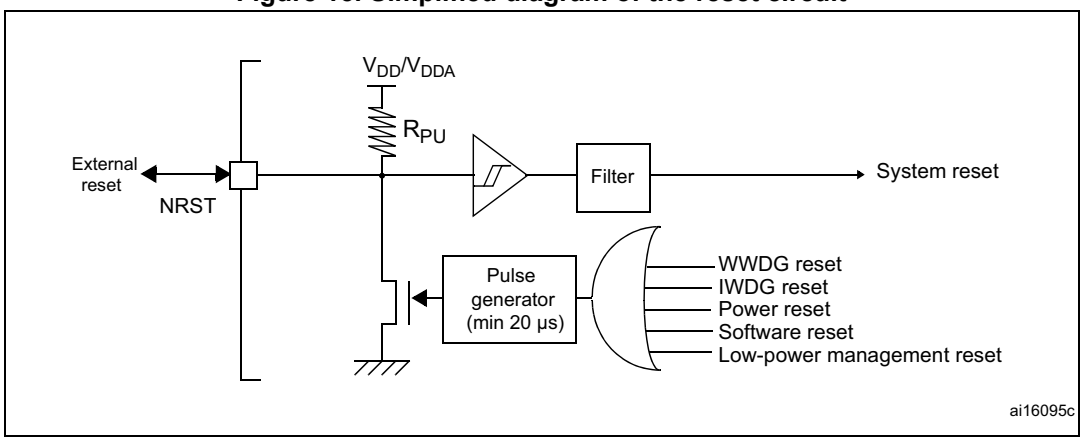
1. Power-on/power-down reset (POR/PDR reset) or brownout (BOR) reset
2. When exiting the Standby mode

A power reset sets all registers to their reset values except the Backup domain (see *Figure 15*)

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

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

Figure 15. Simplified diagram of the reset circuit



The Backup domain has two specific resets that affect only the Backup domain (see *Figure 15*).

6.1.3 Backup domain reset

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

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

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

6.2 Clocks

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

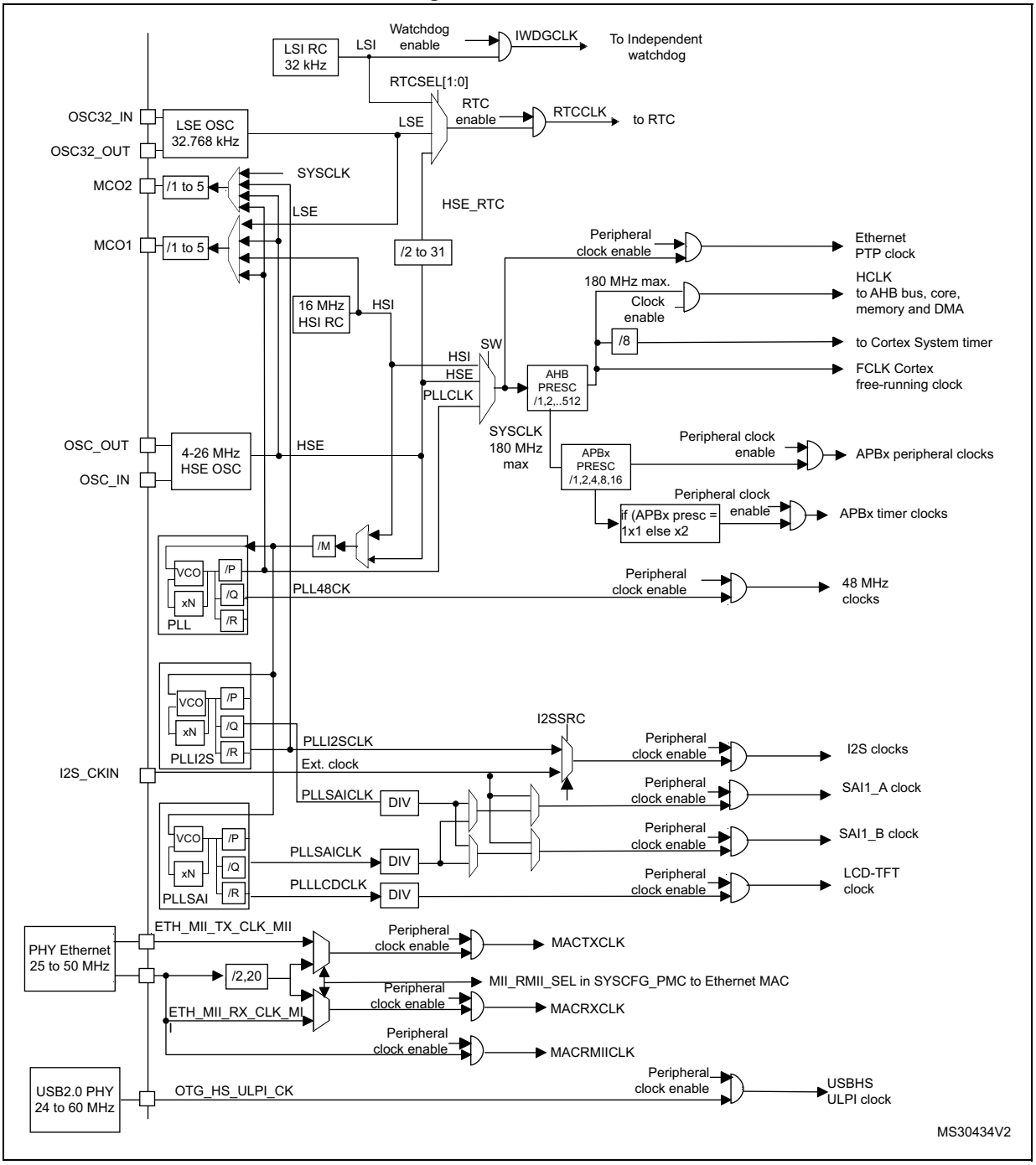
- HSI oscillator clock
- HSE oscillator clock
- Main PLL (PLL) clock

The devices have the two following secondary clock sources:

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

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

Figure 16. Clock tree



1. For full details about the internal and external clock source characteristics, refer to the Electrical characteristics section in the device datasheet.
2. When TIMPRE bit of the RCC_DCKCFGR register is reset, if APBx prescaler is 1, then TIMxCLK = PCLKx, otherwise TIMxCLK = 2x PCLKx.
3. When TIMPRE bit in the RCC_DCKCFGR register is set, if APBx prescaler is 1,2 or 4, then TIMxCLK = HCLK, otherwise TIMxCLK = 4x PCLKx.

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

Several prescalers are used to configure the AHB frequency, the high-speed APB (APB2) and the low-speed APB (APB1) domains. The maximum frequency of the AHB domain is 180 MHz. The maximum allowed frequency of the high-speed APB2 domain is 90 MHz. The maximum allowed frequency of the low-speed APB1 domain is 45 MHz

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

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

  To achieve high-quality audio performance, the I2S clock can be derived either from a specific PLL (PLLI2S) or from an external clock mapped on the I2S_CKIN pin. For more information about I2S clock frequency and precision, refer to *Section 28.4.4: Clock generator*.
- SAI1 clock

  The SAI1 clock is generated from a specific PLL (PLLSAI or PLLI2S) or from an external clock mapped on the I2S_CKIN pin.

  The PLLSAI can be used as clock source for SAI1 peripheral in case the PLLI2S is programmed to achieve another audio sampling frequency (49.152 MHz or 11.2896 MHz), and the application requires both frequencies at the same time.
- LTDC clock

  The LTDC clock is generated from a specific PLL (PLLSAI).
- The USB OTG HS (60 MHz) clock which is provided from the external PHY
- The Ethernet MAC clocks (TX, RX and RMII) which are provided from the external PHY. For further information on the Ethernet configuration, please refer to *Section 33.4.4: MII/RMII selection* in the Ethernet peripheral description. When the Ethernet is used, the AHB clock frequency must be at least 25 MHz.

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

The timer clock frequencies for STM32F42xxx and STM32F43xxx are automatically set by hardware. There are two cases depending on the value of TIMPRE bit in RCC_CFGR register:

- If TIMPRE bit in RCC_DKCFGR register is reset:

  If the APB prescaler is configured to a division factor of 1, the timer clock frequencies (TIMxCLK) are set to PCLKx. Otherwise, the timer clock frequencies are twice the frequency of the APB domain to which the timers are connected: TIMxCLK = 2xPCLKx.
- If TIMPRE bit in RCC_DKCFGR register is set:

  If the APB prescaler is configured to a division factor of 1, 2 or 4, the timer clock frequencies (TIMxCLK) are set to HCLK. Otherwise, the timer clock frequencies is four times the frequency of the APB domain to which the timers are connected: TIMxCLK = 4xPCLKx.

FCLK acts as Cortex®-M4 with FPU free-running clock. For more details, refer to the Cortex®-M4 with FPU technical reference manual.

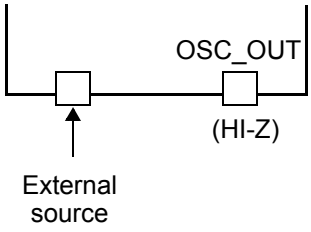
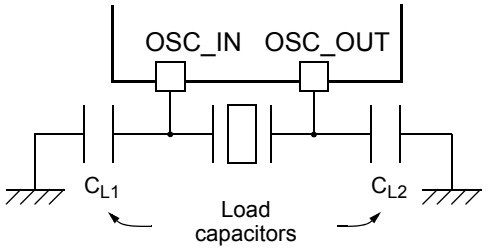
6.2.1 HSE clock

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

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

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

Figure 17. HSE/ LSE clock sources

	Hardware configuration
External clock	
Crystal/ceramic resonators	

External source (HSE bypass)

In this mode, an external clock source must be provided. You select this mode by setting the HSEBYP and HSEON bits in the *RCC clock control register (RCC_CR)*. The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC_IN pin while the OSC_OUT pin should be left HI-Z. See *Figure 17*.

External crystal/ceramic resonator (HSE crystal)

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

The associated hardware configuration is shown in *Figure 17*. Refer to the electrical characteristics section of the *datasheet* for more details.

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

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

6.2.2 HSI clock

The HSI clock signal is generated from an internal 16 MHz RC oscillator and can be used directly as a system clock, or used as PLL input.

The HSI RC oscillator has the advantage of providing a clock source at low cost (no external components). It also has a faster startup time than the HSE crystal oscillator however, even with calibration the frequency is less accurate than an external crystal oscillator or ceramic resonator.

Calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by ST for 1% accuracy at T_A= 25 °C.

After reset, the factory calibration value is loaded in the HSICAL[7:0] bits in the *RCC clock control register (RCC_CR)*.

If the application is subject to voltage or temperature variations this may affect the RC oscillator speed. You can trim the HSI frequency in the application using the HSITRIM[4:0] bits in the *RCC clock control register (RCC_CR)*.

The HSIRDY flag in the *RCC clock control register (RCC_CR)* indicates if the HSI RC is stable or not. At startup, the HSI RC output clock is not released until this bit is set by hardware.

The HSI RC can be switched on and off using the HSION bit in the *RCC clock control register (RCC_CR)*.

The HSI signal can also be used as a backup source (Auxiliary clock) if the HSE crystal oscillator fails. Refer to *Section 6.2.7: Clock security system (CSS) on page 157*.

6.2.3 PLL configuration

The STM32F4xx devices feature three PLLs:

- A main PLL (PLL) clocked by the HSE or HSI oscillator and featuring two different output clocks:
  - The first output is used to generate the high speed system clock (up to 180 MHz)
  - The second output is used to generate the clock for the USB OTG FS (48 MHz), the random analog generator (≤48 MHz) and the SDIO (≤48 MHz).
- Two dedicated PLLs (PLLI2S and PLLSAI) used to generate an accurate clock to achieve high-quality audio performance on the I2S and SAI1 interfaces. PLLSAI is also used for the LCD-TFT clock.

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

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

The three PLLs are disabled by hardware when entering Stop and Standby modes, or when an HSE failure occurs when HSE or PLL (clocked by HSE) are used as system clock. *RCC PLL configuration register (RCC_PLLCFGR)*,*RCC clock configuration register (RCC_CFGR)*, and *RCC Dedicated Clock Configuration Register (RCC_DCKCFGR)* can be used to configure PLL, PLLI2S, and PLLSAI.

6.2.4 LSE clock

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

The LSE oscillator is switched on and off using the LSEON bit in *RCC Backup domain control register (RCC_BDCR)*.

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

External source (LSE bypass)

In this mode, an external clock source must be provided. It must have a frequency up to 1 MHz. You select this mode by setting the LSEBYP and LSEON bits in the *RCC Backup domain control register (RCC_BDCR)*. The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC32_IN pin while the OSC32_OUT pin should be left HI-Z. See *Figure 17*.

6.2.5 LSI clock

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

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

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

6.2.6 System clock (SYSCLK) selection

After a system reset, the HSI oscillator is selected as the system clock. When a clock source is used directly or through PLL as the system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source that is not yet ready is selected, the switch occurs when the clock source is ready. Status bits in the *RCC clock control register (RCC_CR)* indicate which clock(s) is (are) ready and which clock is currently used as the system clock.

6.2.7 Clock security system (CSS)

The clock security system can be activated by software. In this case, the clock detector is enabled after the HSE oscillator startup delay, and disabled when this oscillator is stopped.

If a failure is detected on the HSE clock, this oscillator is automatically disabled, a clock failure event is sent to the break inputs of advanced-control timers TIM1 and TIM8, and an interrupt is generated to inform the software about the failure (clock security system interrupt CSSI), allowing the MCU to perform rescue operations. The CSSI is linked to the Cortex®-M4 with FPU NMI (non-maskable interrupt) exception vector.

Note: When the CSS is enabled, if the HSE clock happens to fail, the CSS generates an interrupt, which causes the automatic generation of an NMI. The NMI is executed indefinitely unless the CSS interrupt pending bit is cleared. As a consequence, the application has to clear the CSS interrupt in the NMI ISR by setting the CSSC bit in the Clock interrupt register (RCC_CIR).

If the HSE oscillator is used directly or indirectly as the system clock (indirectly meaning that it is directly used as PLL input clock, and that PLL clock is the system clock) and a failure is detected, then the system clock switches to the HSI oscillator and the HSE oscillator is disabled.

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

6.2.8 RTC/AWU clock

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

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

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

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

- If LSE is selected as the RTC clock:
  - The RTC continues to work even if the V_{DD} supply is switched off, provided the V_{BAT} supply is maintained.
- If LSI is selected as the Auto-wakeup unit (AWU) clock:
  - The AWU state is not guaranteed if the V_{DD} supply is powered off. Refer to *Section 6.2.5: LSI clock on page 156* for more details on LSI calibration.
- If the HSE clock is used as the RTC clock:
  - The RTC state is not guaranteed if the V_{DD} supply is powered off or if the internal voltage regulator is powered off (removing power from the 1.2 V domain).

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

6.2.9 Watchdog clock

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

6.2.10 Clock-out capability

Two microcontroller clock output (MCO) pins are available:

- MCO1

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

  – HSI clock
  – LSE clock
  – HSE clock
  – PLL clock

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

- MCO2

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

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

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

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

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

6.2.11 Internal/external clock measurement using TIM5/TIM11

It is possible to indirectly measure the frequencies of all on-board clock source generators by means of the input capture of TIM5 channel4 and TIM11 channel1 as shown in *Figure 18* and *Figure 19*.

Internal/external clock measurement using TIM5 channel4

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

The primary purpose of having the LSE connected to the channel4 input capture is to be able to precisely measure the HSI (this requires to have the HSI used as the system clock source). The number of HSI clock counts between consecutive edges of the LSE signal provides a measurement of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppm) we can determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process and/or temperature- and voltage-related frequency deviations.

The HSI oscillator has dedicated, user-accessible calibration bits for this purpose.

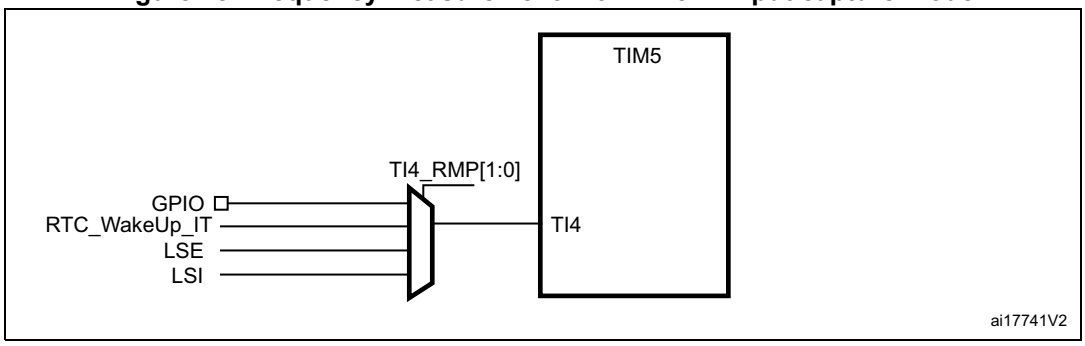
The basic concept consists in providing a relative measurement (e.g. HSI/LSE ratio): the precision is therefore tightly linked to the ratio between the two clock sources. The greater the ratio, the better the measurement.

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

Use the following procedure to measure the LSI frequency:

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

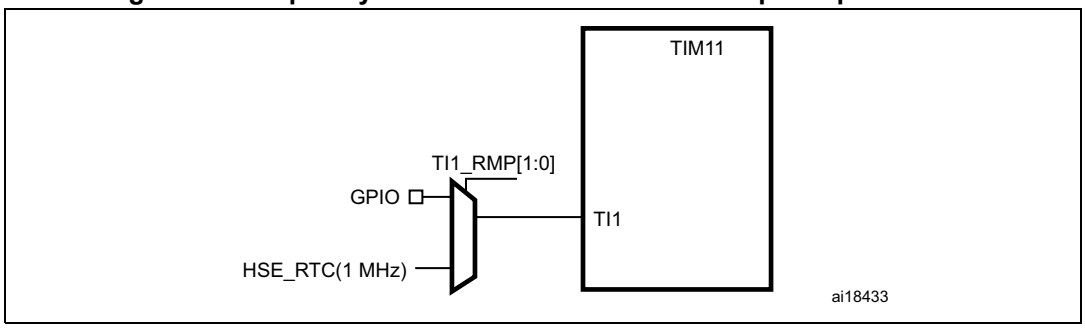
Figure 18. Frequency measurement with TIM5 in Input capture mode



Internal/external clock measurement using TIM11 channel1

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

Figure 19. Frequency measurement with TIM11 in Input capture mode



6.3 RCC registers

Refer to *Section 1.1: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

6.3.1 RCC clock control register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 XX83 where X is undefined.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		PLLSAIRDY	PLLSAION	PLLI2SRDY	PLLI2SON	PLLRDY	PLLON	Reserved				CSSON	HSEBYP	HSERDY	HSEON
		r	rw	r	rw	r	rw					rw	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res.	HSIRDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

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

Bit 29 **PLLSAIRDY**: PLLSAI clock ready flag
Set by hardware to indicate that the PLLSAI is locked.
0: PLLSAI unlocked
1: PLLSAI locked

Bit 28 **PLLSAION**: PLLSAI enable
Set and cleared by software to enable PLLSAI.
Cleared by hardware when entering Stop or Standby mode.
0: PLLSAI OFF
1: PLLSAI ON

Bit 27 **PLLI2SRDY**: PLLI2S clock ready flag
Set by hardware to indicate that the PLLI2S is locked.
0: PLLI2S unlocked
1: PLLI2S locked

Bit 26 **PLLI2SON**: PLLI2S enable
Set and cleared by software to enable PLLI2S.
Cleared by hardware when entering Stop or Standby mode.
0: PLLI2S OFF
1: PLLI2S ON

Bit 25 **PLLRDY**: Main PLL (PLL) clock ready flag
Set by hardware to indicate that PLL is locked.
0: PLL unlocked
1: PLL locked

Bit 24 **PLLON**: Main PLL (PLL) enable
Set and cleared by software to enable PLL.
Cleared by hardware when entering Stop or Standby mode. This bit cannot be reset if PLL clock is used as the system clock.
0: PLL OFF
1: PLL ON

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **CSSON**: Clock security system enable

Set and cleared by software to enable the clock security system. When CSSON is set, the clock detector is enabled by hardware when the HSE oscillator is ready, and disabled by hardware if an oscillator failure is detected.
0: Clock security system OFF (Clock detector OFF)
1: Clock security system ON (Clock detector ON if HSE oscillator is stable, OFF if not)

Bit 18 **HSEBYP**: HSE clock bypass

Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit, to be used by the device.
The HSEBYP bit can be written only if the HSE oscillator is disabled.
0: HSE oscillator not bypassed
1: HSE oscillator bypassed with an external clock

Bit 17 **HSERDY**: HSE clock ready flag

Set by hardware to indicate that the HSE oscillator is stable. After the HSEON bit is cleared, HSERDY goes low after 6 HSE oscillator clock cycles.
0: HSE oscillator not ready
1: HSE oscillator ready

Bit 16 **HSEON**: HSE clock enable

Set and cleared by software.
Cleared by hardware to stop the HSE oscillator when entering Stop or Standby mode. This bit cannot be reset if the HSE oscillator is used directly or indirectly as the system clock.
0: HSE oscillator OFF
1: HSE oscillator ON

Bits 15:8 **HSICAL[7:0]**: Internal high-speed clock calibration

These bits are initialized automatically at startup.

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

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

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HSIRDY**: Internal high-speed clock ready flag

Set by hardware to indicate that the HSI oscillator is stable. After the HSION bit is cleared, HSIRDY goes low after 6 HSI clock cycles.
0: HSI oscillator not ready
1: HSI oscillator ready

Bit 0 **HSION**: Internal high-speed clock enable

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

6.3.2 RCC PLL configuration register (RCC_PLLCFGR)

Address offset: 0x04

Reset value: 0x2400 3010

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

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				PLLQ3	PLLQ2	PLLQ1	PLLQ0	Reserved	PLLSRC	Reserved				PLLP1	PLLP0
				rw	rw	rw	rw		rw					rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLLN									PLLM5	PLLM4	PLLM3	PLLM2	PLLM1	PLLM0
	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 27:24 **PLLQ:** Main PLL (PLL) division factor for USB OTG FS, SDIO and random number generator clocks

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

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

USB OTG FS clock frequency = VCO frequency / PLLQ with 2 ≤PLLQ ≤15

0000: PLLQ = 0, wrong configuration

0001: PLLQ = 1, wrong configuration

0010: PLLQ = 2

0011: PLLQ = 3

0100: PLLQ = 4

...

1111: PLLQ = 15

Bit 23 Reserved, must be kept at reset value.

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

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

0: HSI clock selected as PLL and PLLI2S clock entry

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

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

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

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

Caution: The software has to set these bits correctly not to exceed 180 MHz on this domain.

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

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

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

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

Caution: The software has to set these bits correctly to ensure that the VCO output frequency is between 100 and 432 MHz.

VCO output frequency = VCO input frequency × PLLN with 50 ≤PLLN ≤432

000000000: PLLN = 0, wrong configuration
000000001: PLLN = 1, wrong configuration
...
000110010: PLLN = 50
...
001100011: PLLN = 99
001100100: PLLN = 100
...
110110000: PLLN = 432
110110001: PLLN = 433, wrong configuration
...
111111111: PLLN = 511, wrong configuration

Note: Multiplication factors ranging from 50 and 99 are possible for VCO input frequency higher than 1 MHz. However care must be taken that the minimum VCO output frequency respects the value specified above.

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

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

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

VCO input frequency = PLL input clock frequency / PLLM with 2 ≤PLLM ≤63

000000: PLLM = 0, wrong configuration
000001: PLLM = 1, wrong configuration
000010: PLLM = 2
000011: PLLM = 3
000100: PLLM = 4
...
111110: PLLM = 62
111111: PLLM = 63

6.3.3 RCC clock configuration register (RCC_CFGR)

Address offset: 0x08

Reset value: 0x0000 0000

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2		MCO2 PRE[2:0]			MCO1 PRE[2:0]			I2SSCR	MCO1		RTCPRE[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
PPRE2[2:0]			PPRE1[2:0]			Reserved		HPRE[3:0]				SWS1	SWS0	SW1	SW0
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	r	r	rw	rw

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

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

00: System clock (SYSCLK) selected

01: PLLI2S clock selected

10: HSE oscillator clock selected

11: PLL clock selected

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

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

0xx: no division

100: division by 2

101: division by 3

110: division by 4

111: division by 5

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

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

0xx: no division

100: division by 2

101: division by 3

110: division by 4

111: division by 5

Bit 23 **I2SSRC**: I2S clock selection

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

0: PLLI2S clock used as I2S clock source

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

Bits 22:21 **MCO1:** Microcontroller clock output 1

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

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

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

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

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

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

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

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

Caution: The software has to set these bits correctly not to exceed 90 MHz on this domain. The clocks are divided with the new prescaler factor from 1 to 16 AHB cycles after PPRE2 write.

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

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

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

Caution: The software has to set these bits correctly not to exceed 45 MHz on this domain. The clocks are divided with the new prescaler factor from 1 to 16 AHB cycles after PPRE1 write.

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

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

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

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

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

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

0xxx: system clock not divided
1000: system clock divided by 2
1001: system clock divided by 4
1010: system clock divided by 8
1011: system clock divided by 16
1100: system clock divided by 64
1101: system clock divided by 128
1110: system clock divided by 256
1111: system clock divided by 512

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

Set and cleared by hardware to indicate which clock source is used as the system clock.
00: HSI oscillator used as the system clock
01: HSE oscillator used as the system clock
10: PLL used as the system clock
11: not applicable

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

Set and cleared by software to select the system clock source.
Set by hardware to force the HSI selection when leaving the Stop or Standby mode or in case of failure of the HSE oscillator used directly or indirectly as the system clock.
00: HSI oscillator selected as system clock
01: HSE oscillator selected as system clock
10: PLL selected as system clock
11: not allowed

6.3.4 RCC clock interrupt register (RCC_CIR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CSSC	PLLSAI RDYC	PLLI2S RDYC	PLL RDYC	HSE RDYC	HSI RDYC	LSE RDYC	LSI RDYC
								w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLLSAI RDYIE	PLLI2S RDYIE	PLL RDYIE	HSE RDYIE	HSI RDYIE	LSE RDYIE	LSI RDYIE	CSSF	PLLSAI RDYF	PLLI2S RDYF	PLL RDYF	HSE RDYF	HSI RDYF	LSE RDYF	LSI RDYF
	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r

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

Bit 23 **CSSC:** Clock security system interrupt clear
This bit is set by software to clear the CSSF flag.
0: No effect
1: Clear CSSF flag

Bit 22 **PLLSAIRDYC**: PLLSAI Ready Interrupt Clear
This bit is set by software to clear PLLSAIRDYF flag. It is reset by hardware when the PLLSAIRDYF is cleared.
0: PLLSAIRDYF not cleared
1: PLLSAIRDYF cleared

Bit 21 **PLLI2SRDYC:** PLLI2S ready interrupt clear
This bit is set by software to clear the PLLI2SRDYF flag.
0: No effect
1: PLLI2SRDYF cleared

Bit 20 **PLLRDYC:** Main PLL(PLL) ready interrupt clear
This bit is set by software to clear the PLLRDYF flag.
0: No effect
1: PLLRDYF cleared

Bit 19 **HSERDYC:** HSE ready interrupt clear
This bit is set by software to clear the HSERDYF flag.
0: No effect
1: HSERDYF cleared

Bit 18 **HSIRDYC:** HSI ready interrupt clear
This bit is set software to clear the HSIRDYF flag.
0: No effect
1: HSIRDYF cleared

Bit 17 **LSERDYC:** LSE ready interrupt clear
This bit is set by software to clear the LSERDYF flag.
0: No effect
1: LSERDYF cleared

Bit 16 **LSIRDYC:** LSI ready interrupt clear
This bit is set by software to clear the LSIRDYF flag.
0: No effect
1: LSIRDYF cleared

Bit 15 Reserved, must be kept at reset value.

Bit 14 **PLLSAIRDYIE**: PLLSAI Ready Interrupt Enable
This bit is set and reset by software to enable/disable interrupt caused by PLLSAI lock.
0: PLLSAI lock interrupt disabled
1: PLLSAI lock interrupt enabled

Bit 13 **PLLI2SRDYIE:** PLLI2S ready interrupt enable
This bit is set and cleared by software to enable/disable interrupt caused by PLLI2S lock.
0: PLLI2S lock interrupt disabled
1: PLLI2S lock interrupt enabled

Bit 12 **PLLRDYIE:** Main PLL (PLL) ready interrupt enable
This bit is set and cleared by software to enable/disable interrupt caused by PLL lock.
0: PLL lock interrupt disabled
1: PLL lock interrupt enabled

Bit 11 **HSERDYIE:** HSE ready interrupt enable
This bit is set and cleared by software to enable/disable interrupt caused by the HSE oscillator stabilization.
0: HSE ready interrupt disabled
1: HSE ready interrupt enabled

Bit 10 **HSIRDYIE:** HSI ready interrupt enable
This bit is set and cleared by software to enable/disable interrupt caused by the HSI oscillator stabilization.
0: HSI ready interrupt disabled
1: HSI ready interrupt enabled

Bit 9 **LSERDYIE:** LSE ready interrupt enable
This bit is set and cleared by software to enable/disable interrupt caused by the LSE oscillator stabilization.
0: LSE ready interrupt disabled
1: LSE ready interrupt enabled

Bit 8 **LSIRDYIE:** LSI ready interrupt enable
This bit is set and cleared by software to enable/disable interrupt caused by LSI oscillator stabilization.
0: LSI ready interrupt disabled
1: LSI ready interrupt enabled

Bit 7 **CSSF:** Clock security system interrupt flag
This bit is set by hardware when a failure is detected in the HSE oscillator.
It is cleared by software by setting the CSSC bit.
0: No clock security interrupt caused by HSE clock failure
1: Clock security interrupt caused by HSE clock failure

Bit 6 **PLLSAIRDYF**: PLLSAI Ready Interrupt flag
This bit is set by hardware when the PLLSAI is locked and PLLSAIRDYDIE is set.
It is cleared by software by setting the PLLSAIRDYC bit.
0: No clock ready interrupt caused by PLLSAI lock
1: Clock ready interrupt caused by PLLSAI lock

Bit 5 **PLLI2SRDYF:** PLLI2S ready interrupt flag
This bit is set by hardware when the PLLI2S is locked and PLLI2SRDYDIE is set.
It is cleared by software by setting the PLLRI2SDYC bit.
0: No clock ready interrupt caused by PLLI2S lock
1: Clock ready interrupt caused by PLLI2S lock

Bit 4 **PLLRDYF:** Main PLL (PLL) ready interrupt flag
This bit is set by hardware when PLL is locked and PLLRDYDIE is set.
It is cleared by software setting the PLLRDYC bit.
0: No clock ready interrupt caused by PLL lock
1: Clock ready interrupt caused by PLL lock

Bit 3 **HSERDYF:** HSE ready interrupt flag

This bit is set by hardware when External High Speed clock becomes stable and HSERDYDIE is set.

It is cleared by software by setting the HSERDYC bit.

0: No clock ready interrupt caused by the HSE oscillator

1: Clock ready interrupt caused by the HSE oscillator

Bit 2 **HSIRDYF:** HSI ready interrupt flag

This bit is set by hardware when the Internal High Speed clock becomes stable and HSIRDYDIE is set.

It is cleared by software by setting the HSIRDYC bit.

0: No clock ready interrupt caused by the HSI oscillator

1: Clock ready interrupt caused by the HSI oscillator

Bit 1 **LSERDYF:** LSE ready interrupt flag

This bit is set by hardware when the External Low Speed clock becomes stable and LSERDYDIE is set.

It is cleared by software by 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

This bit is set by hardware when the internal low speed clock becomes stable and LSIRDYDIE is set.

It is cleared by software by setting the LSIRDYC bit.

0: No clock ready interrupt caused by the LSI oscillator

1: Clock ready interrupt caused by the LSI oscillator

6.3.5 RCC AHB1 peripheral reset register (RCC_AHB1RSTR)

Address offset: 0x10

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		OTGHS RST	Reserved			ETHMAC RST	Res.	DMA2D RST	DMA2 RST	DMA1 RST	Reserved				
		rw				rw		rw	rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CRCRST	Res.	GPIOKRST	GPIOJRST	GPIOIRST	GPIOHRST	GPIOGGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

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

This bit is set and cleared by software.

0: does not reset the USB OTG HS module

1: resets the USB OTG HS module

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

Bit 25 **ETHMACRST:** Ethernet MAC reset
This bit is set and cleared by software.
0: does not reset Ethernet MAC
1: resets Ethernet MAC

Bit 24 Reserved, must be kept at reset value.

Bit 23 **DMA2DRST**: DMA2D reset
This bit is set and reset by software.
0: does not reset DMA2D
1: resets DMA2D

Bit 22 **DMA2RST:** DMA2 reset
This bit is set and cleared by software.
0: does not reset DMA2
1: resets DMA2

Bit 21 **DMA1RST:** DMA2 reset
This bit is set and cleared by software.
0: does not reset DMA2
1: resets DMA2

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

Bit 12 **CRCRST:** CRC reset
This bit is set and cleared by software.
0: does not reset CRC
1: resets CRC

Bit 11 Reserved, must be kept at reset value.

Bit 10 **GPIOKRST**: IO port K reset
This bit is set and cleared by software.
0: does not reset IO port K
1: resets IO port K

Bit 9 **GPIOJRST**: IO port J reset
This bit is set and cleared by software.
0: does not reset IO port J
1: resets IO port J

Bit 8 **GPIOIRST:** IO port I reset
This bit is set and cleared by software.
0: does not reset IO port I
1: resets IO port I

Bit 7 **GPIOHRST:** IO port H reset
This bit is set and cleared by software.
0: does not reset IO port H
1: resets IO port H

Bit 6 **GPIOGRST:** IO port G reset
This bit is set and cleared by software.
0: does not reset IO port G
1: resets IO port G

Bit 5 **GPIOFRST:** IO port F reset
This bit is set and cleared by software.
0: does not reset IO port F
1: resets IO port F

Bit 4 **GPIOERST:** IO port E reset
This bit is set and cleared by software.
0: does not reset IO port E
1: resets IO port E

Bit 3 **GPIODRST:** IO port D reset
This bit is set and cleared by software.
0: does not reset IO port D
1: resets IO port D

Bit 2 **GPIOCRST:** IO port C reset
This bit is set and cleared by software.
0: does not reset IO port C
1: resets IO port C

Bit 1 **GPIOBRST:** IO port B reset
This bit is set and cleared by software.
0: does not reset IO port B
1:resets IO port B

Bit 0 **GPIOARST:** IO port A reset
This bit is set and cleared by software.
0: does not reset IO port A
1: resets IO port A

6.3.6 RCC AHB2 peripheral reset register (RCC_AHB2RSTR)

Address offset: 0x14

Reset value: 0x0000 0000

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

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

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

Bit 7 **OTGFSRST:** USB OTG FS module reset
Set and cleared by software.
0: does not reset the USB OTG FS module
1: resets the USB OTG FS module

Bit 6 **RNGRST:** Random number generator module reset
Set and cleared by software.
0: does not reset the random number generator module
1: resets the random number generator module

Bit 5 **HASHRST:** Hash module reset
Set and cleared by software.
0: does not reset the HASH module
1: resets the HASH module

Bit 4 **CRYPRST:** Cryptographic module reset
Set and cleared by software.
0: does not reset the cryptographic module
1: resets the cryptographic module

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

Bit 0 **DCMIRST:** Camera interface reset
Set and cleared by software.
0: does not reset the Camera interface
1: resets the Camera interface

6.3.7 RCC AHB3 peripheral reset register (RCC_AHB3RSTR)

Address offset: 0x18

Reset value: 0x0000 0000

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

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

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

Bit 0 **FMCRST:** Flexible memory controller module reset
Set and cleared by software.
0: does not reset the FMC module
1: resets the FMC module

6.3.8 RCC APB1 peripheral reset register (RCC_APB1RSTR)

Address offset: 0x20

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8RST	UART7RST	DACRST	PWRRST	Reserved	CAN2RST	CAN1RST	Reserved	I2C3RST	I2C2RST	I2C1RST	UART5RST	UART4RST	UART3RST	UART2RST	Reserved
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	Reserved		WWDGRST	Reserved		TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UART8RST**: UART8 reset
Set and cleared by software.
0: does not reset UART8
1: resets UART8

Bit 30 **UART7RST**: UART7 reset
Set and cleared by software.
0: does not reset UART7
1: resets UART7

Bit 29 **DACRST:** DAC reset
Set and cleared by software.
0: does not reset the DAC interface
1: resets the DAC interface

Bit 28 **PWRRST:** Power interface reset
Set and cleared by software.
0: does not reset the power interface
1: resets the power interface

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CAN2RST:** CAN2 reset
Set and cleared by software.
0: does not reset CAN2
1: resets CAN2

Bit 25 **CAN1RST:** CAN1 reset
Set and cleared by software.
0: does not reset CAN1
1: resets CAN1

Bit 24 Reserved, must be kept at reset value.

Bit 23 **I2C3RST:** I2C3 reset
Set and cleared by software.
0: does not reset I2C3
1: resets I2C3

Bit 22 **I2C2RST:** I2C2 reset
Set and cleared by software.
0: does not reset I2C2
1: resets I2C2

Bit 21 **I2C1RST:** I2C1 reset
Set and cleared by software.
0: does not reset I2C1
1: resets I2C1

Bit 20 **UART5RST:** UART5 reset
Set and cleared by software.
0: does not reset UART5
1: resets UART5

Bit 19 **UART4RST:** USART4 reset
Set and cleared by software.
0: does not reset UART4
1: resets UART4

Bit 18 **USART3RST:** USART3 reset
Set and cleared by software.
0: does not reset USART3
1: resets USART3

Bit 17 **USART2RST:** USART2 reset
Set and cleared by software.
0: does not reset USART2
1: resets USART2

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3RST:** SPI3 reset
Set and cleared by software.
0: does not reset SPI3
1: resets SPI3

Bit 14 **SPI2RST:** SPI2 reset
Set and cleared by software.
0: does not reset SPI2
1: resets SPI2

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

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

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

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

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

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

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

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

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

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

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

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

6.3.9 RCC APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x24

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					LTDCRST	Reserved			SAI1RST	SPI6RST	SPI5RST	Res.	TIM11RST	TIM10RST	TIM9RST
					rw				rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFGRST	SPI4RST	SPI1RST	SDIORST	Reserved		ADCRST	Reserved		USART6RST	USART1RST	Reserved		TIM8RST	TIM1RST
	rw	rw	rw	rw			rw			rw	rw			rw	rw

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

Bit 26 **LTDCRST**: LTDC reset
This bit is set and reset by software.
0: does not reset LCD-TFT
1: resets LCD-TFT

Bits 27:23 Reserved, must be kept at reset value.

Bit 22 **SAI1RST**: SAI1 reset
This bit is set and reset by software.
0: does not reset SAI1
1: resets SAI1

Bit 21 **SPI6RST**: SPI6 reset
This bit is set and cleared by software.
0: does not reset SPI6
1: resets SPI6

Bit 20 **SPI5RST**: SPI5 reset
This bit is set and cleared by software.
0: does not reset SPI5
1: resets SPI5

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TIM11RST:** TIM11 reset
This bit is set and cleared by software.
0: does not reset TIM11
1: resets TIM14

Bit 17 **TIM10RST:** TIM10 reset
This bit is set and cleared by software.
0: does not reset TIM10
1: resets TIM10

Bit 16 **TIM9RST:** TIM9 reset
This bit is set and cleared by software.
0: does not reset TIM9
1: resets TIM9

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SYSCFGRST:** System configuration controller reset
This bit is set and cleared by software.
0: does not reset the System configuration controller
1: resets the System configuration controller

Bit 13 **SPI4RST**: SPI4 reset
This bit is set and cleared by software.
0: does not reset SPI4
1: resets SPI4

Bit 12 **SPI1RST:** SPI1 reset
This bit is set and cleared by software.
0: does not reset SPI1
1: resets SPI1

Bit 11 **SDIORST:** SDIO reset
This bit is set and cleared by software.
0: does not reset the SDIO module
1: resets the SDIO module

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

Bit 8 **ADCRST:** ADC interface reset (common to all ADCs)
This bit is set and cleared by software.
0: does not reset the ADC interface
1: resets the ADC interface

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

Bit 5 **USART6RST:** USART6 reset
This bit is set and cleared by software.
0: does not reset USART6
1: resets USART6

Bit 4 **USART1RST:** USART1 reset
This bit is set and cleared by software.
0: does not reset USART1
1: resets USART1

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

Bit 1 **TIM8RST:** TIM8 reset
This bit is set and cleared by software.
0: does not reset TIM8
1: resets TIM8

Bit 0 **TIM1RST:** TIM1 reset
This bit is set and cleared by software.
0: does not reset TIM1
1: resets TIM1

6.3.10 RCC AHB1 peripheral clock register (RCC_AHB1ENR)

Address offset: 0x30

Reset value: 0x0010 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reser-ved	OTGHS ULPIEN	OTGHSEN	ETHMACPTPEN	ETHMACRXEN	ETHMACTXEN	ETHMACEN	Res.	**DMA2DEN**	DMA2EN	DMA1EN	CCMDATARAMEN	Res.	BKPSRAMEN	Reserved	
	rw	rw	rw	rw	rw	rw		rw	rw	rw			rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CRCEN	Res.	GPIOKEN	GPIOJEN	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **OTGHSULPIEN:** USB OTG HSULPI clock enable
This bit is set and cleared by software. It must be cleared when the OTG_HS is used in FS mode.
0: USB OTG HS ULPI clock disabled
1: USB OTG HS ULPI clock enabled

Bit 29 **OTGHSEN:** USB OTG HS clock enable
This bit is set and cleared by software.
0: USB OTG HS clock disabled
1: USB OTG HS clock enabled

Bit 28 **ETHMACPTPEN:** Ethernet PTP clock enable
This bit is set and cleared by software.
0: Ethernet PTP clock disabled
1: Ethernet PTP clock enabled

Bit 27 **ETHMACRXEN:** Ethernet Reception clock enable
This bit is set and cleared by software.
0: Ethernet Reception clock disabled
1: Ethernet Reception clock enabled

Bit 26 **ETHMACTXEN:** Ethernet Transmission clock enable
This bit is set and cleared by software.
0: Ethernet Transmission clock disabled
1: Ethernet Transmission clock enabled

Bit 25 **ETHMACEN:** Ethernet MAC clock enable
This bit is set and cleared by software.
0: Ethernet MAC clock disabled
1: Ethernet MAC clock enabled

Bit 24 Reserved, must be kept at reset value.

Bit 23 **DMA2DEN:** DMA2D clock enable
This bit is set and cleared by software.
0: DMA2D clock disabled
1: DMA2D clock enabled

Bit 22 **DMA2EN:** DMA2 clock enable
This bit is set and cleared by software.
0: DMA2 clock disabled
1: DMA2 clock enabled

Bit 21 **DMA1EN:** DMA1 clock enable
This bit is set and cleared by software.
0: DMA1 clock disabled
1: DMA1 clock enabled

Bit 20 **CCMDATARAMEN**: CCM data RAM clock enable
This bit is set and cleared by software.
0: CCM data RAM clock disabled
1: CCM data RAM clock enabled

Bit 19 Reserved, must be kept at reset value.

Bit 18 **BKPSRAMEN:** Backup SRAM interface clock enable
This bit is set and cleared by software.
0: Backup SRAM interface clock disabled
1: Backup SRAM interface clock enabled

Bits 17:13 Reserved, must be kept at reset value.

Bit 12 **CRCEN:** CRC clock enable
This bit is set and cleared by software.
0: CRC clock disabled
1: CRC clock enabled

Bit 11 Reserved, must be kept at reset value.

Bit 10 **GPIOKEN:** IO port K clock enable
This bit is set and cleared by software.
0: IO port K clock disabled
1: IO port K clock enabled

Bit 9 **GPIOJEN:** IO port J clock enable
This bit is set and cleared by software.
0: IO port J clock disabled
1: IO port J clock enabled

Bit 8 **GPIOIEN:** IO port I clock enable
This bit is 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
This bit is 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
This bit is 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
This bit is 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
This bit is 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
This bit is 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
This bit is 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
This bit is 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
This bit is set and cleared by software.
0: IO port A clock disabled
1: IO port A clock enabled

6.3.11 RCC AHB2 peripheral clock enable register (RCC_AHB2ENR)

Address offset: 0x34

Reset value: 0x0000 0000

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

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

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

Bit 7 **OTGFSEN:** USB OTG FS clock enable
This bit is set and cleared by software.
0: USB OTG FS clock disabled
1: USB OTG FS clock enabled

Bit 6 **RNGEN:** Random number generator clock enable
This bit is set and cleared by software.
0: Random number generator clock disabled
1: Random number generator clock enabled

Bit 5 **HASHEN:** Hash modules clock enable
This bit is set and cleared by software.
0: Hash modules clock disabled
1: Hash modules clock enabled

Bit 4 **CRYPEN:** Cryptographic modules clock enable
This bit is set and cleared by software.
0: cryptographic module clock disabled
1: cryptographic module clock enabled

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

Bit 0 **DCMIEN:** Camera interface enable
This bit is set and cleared by software.
0: Camera interface clock disabled
1: Camera interface clock enabled

6.3.12 RCC AHB3 peripheral clock enable register (RCC_AHB3ENR)

Address offset: 0x38

Reset value: 0x0000 0000

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

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

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

Bit 0 **FMCEN:** Flexible memory controller module clock enable
This bit is set and cleared by software.
0: FMC module clock disabled
1: FMC module clock enabled

6.3.13 RCC APB1 peripheral clock enable register (RCC_APB1ENR)

Address offset: 0x40

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 EN	UART7 EN	DAC EN	PWR EN	Reser-ved	CAN2 EN	CAN1 EN	Reser-ved	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART 3 EN	USART 2 EN	Reser-ved
rw	rw	rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Reserved		WWDG EN	Reserved		TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UART8EN**: UART8 clock enable
This bit is set and cleared by software.
0: UART8 clock disabled
1: UART8 clock enabled

Bit 30 **UART7EN**: UART7 clock enable
This bit is set and cleared by software.
0: UART7 clock disabled
1: UART7 clock enabled

Bit 29 **DACEN:** DAC interface clock enable
This bit is set and cleared by software.
0: DAC interface clock disabled
1: DAC interface clock enable

Bit 28 **PWREN:** Power interface clock enable
This bit is set and cleared by software.
0: Power interface clock disabled
1: Power interface clock enable

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CAN2EN:** CAN 2 clock enable
This bit is set and cleared by software.
0: CAN 2 clock disabled
1: CAN 2 clock enabled

Bit 25 **CAN1EN:** CAN 1 clock enable
This bit is set and cleared by software.
0: CAN 1 clock disabled
1: CAN 1 clock enabled

Bit 24 Reserved, must be kept at reset value.

Bit 23 **I2C3EN:** I2C3 clock enable
This bit is set and cleared by software.
0: I2C3 clock disabled
1: I2C3 clock enabled

Bit 22 **I2C2EN:** I2C2 clock enable
This bit is set and cleared by software.
0: I2C2 clock disabled
1: I2C2 clock enabled

Bit 21 **I2C1EN:** I2C1 clock enable
This bit is set and cleared by software.
0: I2C1 clock disabled
1: I2C1 clock enabled

Bit 20 **UART5EN:** UART5 clock enable
This bit is set and cleared by software.
0: UART5 clock disabled
1: UART5 clock enabled

Bit 19 **UART4EN:** UART4 clock enable
This bit is set and cleared by software.
0: UART4 clock disabled
1: UART4 clock enabled

Bit 18 **USART3EN:** USART3 clock enable
This bit is set and cleared by software.
0: USART3 clock disabled
1: USART3 clock enabled

Bit 17 **USART2EN:** USART2 clock enable
This bit is 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
This bit is set and cleared by software.
0: SPI3 clock disabled
1: SPI3 clock enabled

Bit 14 **SPI2EN:** SPI2 clock enable
This bit is 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
This bit is set and cleared by software.
0: Window watchdog clock disabled
1: Window watchdog clock enabled

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

Bit 8 **TIM14EN:** TIM14 clock enable
This bit is set and cleared by software.
0: TIM14 clock disabled
1: TIM14 clock enabled

Bit 7 **TIM13EN:** TIM13 clock enable
This bit is set and cleared by software.
0: TIM13 clock disabled
1: TIM13 clock enabled

Bit 6 **TIM12EN:** TIM12 clock enable
This bit is set and cleared by software.
0: TIM12 clock disabled
1: TIM12 clock enabled

Bit 5 **TIM7EN:** TIM7 clock enable
This bit is set and cleared by software.
0: TIM7 clock disabled
1: TIM7 clock enabled

Bit 4 **TIM6EN:** TIM6 clock enable
This bit is set and cleared by software.
0: TIM6 clock disabled
1: TIM6 clock enabled

Bit 3 **TIM5EN:** TIM5 clock enable
This bit is set and cleared by software.
0: TIM5 clock disabled
1: TIM5 clock enabled

Bit 2 **TIM4EN:** TIM4 clock enable
This bit is set and cleared by software.
0: TIM4 clock disabled
1: TIM4 clock enabled

Bit 1 **TIM3EN:** TIM3 clock enable
This bit is set and cleared by software.
0: TIM3 clock disabled
1: TIM3 clock enabled

Bit 0 **TIM2EN:** TIM2 clock enable
This bit is set and cleared by software.
0: TIM2 clock disabled
1: TIM2 clock enabled

6.3.14 RCC APB2 peripheral clock enable register (RCC_APB2ENR)

Address offset: 0x44

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					LTDC EN	Reserved			SAI1EN	SPI6EN	SPI5EN	Res.	TIM11 EN	TIM10 EN	TIM9 EN
					rw				rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG EN	SPI4EN	SPI1 EN	SDIO EN	ADC3 EN	ADC2 EN	ADC1 EN	Reserved		USART 6 EN	USART 1 EN	Reserved		TIM8 EN	TIM1 EN
	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw

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

Bit 26 **LTDCEN:** LTDC clock enable
This bit is set and cleared by software.
0: LTDC clock disabled
1: LTDC clock enabled

Bits 27: 23 Reserved, must be kept at reset value.

Bit 22 **SAI1EN:** SAI1 clock enable
This bit is set and cleared by software.
0: SAI1 clock disabled
1: SAI1 clock enabled

Bit 21 **SPI6EN:** SPI6 clock enable
This bit is set and cleared by software.
0: SPI6 clock disabled
1: SPI6 clock enabled

Bit 20 **SPI5EN:** SPI5 clock enable
This bit is set and cleared by software.
0: SPI5 clock disabled
1: SPI5 clock enabled

Bit 18 **TIM11EN:** TIM11 clock enable
This bit is set and cleared by software.
0: TIM11 clock disabled
1: TIM11 clock enabled

Bit 17 **TIM10EN:** TIM10 clock enable
This bit is set and cleared by software.
0: TIM10 clock disabled
1: TIM10 clock enabled

Bit 16 **TIM9EN:** TIM9 clock enable
This bit is set and cleared by software.
0: TIM9 clock disabled
1: TIM9 clock enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SYSCFGEN:** System configuration controller clock enable
This bit is set and cleared by software.
0: System configuration controller clock disabled
1: System configuration controller clock enabled

Bit 13 **SPI4EN:** SPI4 clock enable
This bit is set and cleared by software.
0: SPI4 clock disabled
1: SPI4 clock enabled

Bit 12 **SPI1EN:** SPI1 clock enable
This bit is set and cleared by software.
0: SPI1 clock disabled
1: SPI1 clock enabled

Bit 11 **SDIOEN:** SDIO clock enable
This bit is set and cleared by software.
0: SDIO module clock disabled
1: SDIO module clock enabled

Bit 10 **ADC3EN:** ADC3 clock enable
This bit is set and cleared by software.
0: ADC3 clock disabled
1: ADC3 clock enabled

Bit 9 **ADC2EN:** ADC2 clock enable
This bit is set and cleared by software.
0: ADC2 clock disabled
1: ADC2 clock enabled

Bit 8 **ADC1EN:** ADC1 clock enable
This bit is set and cleared by software.
0: ADC1 clock disabled
1: ADC1 clock enabled

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

Bit 5 **USART6EN:** USART6 clock enable
This bit is set and cleared by software.
0: USART6 clock disabled
1: USART6 clock enabled

Bit 4 **USART1EN:** USART1 clock enable
This bit is set and cleared by software.
0: USART1 clock disabled
1: USART1 clock enabled

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

Bit 1 **TIM8EN:** TIM8 clock enable
This bit is set and cleared by software.
0: TIM8 clock disabled
1: TIM8 clock enabled

Bit 0 **TIM1EN:** TIM1 clock enable
This bit is set and cleared by software.
0: TIM1 clock disabled
1: TIM1 clock enabled

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

Address offset: 0x50

Reset value: 0x7EEF 97FF

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.	OTGHS ULPILPEN	OTGHS LPEN	ETHPTP LPEN	ETHRX LPEN	ETHTX LPEN	ETHMAC LPEN	Res.	DMA2D LPEN	DMA2 LPEN	DMA1 LPEN	Res.	SRAM3 LPEN	BKPSRAM LPEN	SRAM2 LPEN	SRAM1 LPEN
	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
FLITF LPEN	Reserved		CRC LPEN	Res.	GPIOK LPEN	GPIOIJ LPEN	GPIOI LPEN	GPIOH LPEN	GPIOG LPEN	GPIOF LPEN	GPIOE LPEN	GPIOD LPEN	GPIOC LPEN	GPIOB LPEN	GPIOA LPEN
rw			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **OTGHSULPILPEN:** USB OTG HS ULPI clock enable during Sleep mode
This bit is set and cleared by software. It must be cleared when the OTG_HS is used in FS mode.
0: USB OTG HS ULPI clock disabled during Sleep mode
1: USB OTG HS ULPI clock enabled during Sleep mode

Bit 29 **OTGHSLPEN:** USB OTG HS clock enable during Sleep mode
This bit is set and cleared by software.
0: USB OTG HS clock disabled during Sleep mode
1: USB OTG HS clock enabled during Sleep mode

Bit 28 **ETHMACPTPLPEN:** Ethernet PTP clock enable during Sleep mode
This bit is set and cleared by software.
0: Ethernet PTP clock disabled during Sleep mode
1: Ethernet PTP clock enabled during Sleep mode

Bit 27 **ETHMACRXLPEN:** Ethernet reception clock enable during Sleep mode
This bit is set and cleared by software.
0: Ethernet reception clock disabled during Sleep mode
1: Ethernet reception clock enabled during Sleep mode

Bit 26 **ETHMACTXLPEN:** Ethernet transmission clock enable during Sleep mode
This bit is set and cleared by software.
0: Ethernet transmission clock disabled during sleep mode
1: Ethernet transmission clock enabled during sleep mode

Bit 25 **ETHMACLPEN:** Ethernet MAC clock enable during Sleep mode
This bit is set and cleared by software.
0: Ethernet MAC clock disabled during Sleep mode
1: Ethernet MAC clock enabled during Sleep mode

Bit 24 Reserved, must be kept at reset value.

Bit 23 **DMA2DLPEN:** DMA2D clock enable during Sleep mode
This bit is set and cleared by software.
0: DMA2D clock disabled during Sleep mode
1: DMA2D clock enabled during Sleep mode

Bit 22 **DMA2LPEN:** DMA2 clock enable during Sleep mode
This bit is set and cleared by software.
0: DMA2 clock disabled during Sleep mode
1: DMA2 clock enabled during Sleep mode

Bit 21 **DMA1LPEN:** DMA1 clock enable during Sleep mode
This bit is set and cleared by software.
0: DMA1 clock disabled during Sleep mode
1: DMA1 clock enabled during Sleep mode

Bit 20 Reserved, must be kept at reset value.

Bit 19 **SRAM3LPEN:** SRAM3 interface clock enable during Sleep mode
This bit is set and cleared by software.
0: SRAM3 interface clock disabled during Sleep mode
1: SRAM3 interface clock enabled during Sleep mode

Bit 18 **BKPSRAMLPEN:** Backup SRAM interface clock enable during Sleep mode
This bit is set and cleared by software.
0: Backup SRAM interface clock disabled during Sleep mode
1: Backup SRAM interface clock enabled during Sleep mode

Bit 17 **SRAM2LPEN:** SRAM2 interface clock enable during Sleep mode
This bit is set and cleared by software.
0: SRAM2 interface clock disabled during Sleep mode
1: SRAM2 interface clock enabled during Sleep mode

Bit 16 **SRAM1LPEN:** SRAM1 interface clock enable during Sleep mode
This bit is set and cleared by software.
0: SRAM1 interface clock disabled during Sleep mode
1: SRAM1 interface clock enabled during Sleep mode

Bit 15 **FLITFLPEN:** Flash interface clock enable during Sleep mode
This bit is set and cleared by software.
0: Flash interface clock disabled during Sleep mode
1: Flash interface clock enabled during Sleep mode

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

Bit 12 **CRCLPEN:** CRC clock enable during Sleep mode
This bit is set and cleared by software.
0: CRC clock disabled during Sleep mode
1: CRC clock enabled during Sleep mode

Bit 11 Reserved, must be kept at reset value.

Bit 10 **GPIOKLPEN:** IO port K clock enable during Sleep mode
This bit is set and cleared by software.
0: IO port K clock disabled during Sleep mode
1: IO port K clock enabled during Sleep mode

Bit 9 **GPIOJLPEN:** IO port J clock enable during Sleep mode
This bit is set and cleared by software.
0: IO port J clock disabled during Sleep mode
1: IO port J clock enabled during Sleep mode

Bit 8 **GPIOILPEN:** IO port I clock enable during Sleep mode
This bit is set and cleared by software.
0: IO port I clock disabled during Sleep mode
1: IO port I clock enabled during Sleep mode

Bit 7 **GPIOHLPEN:** IO port H clock enable during Sleep mode
This bit is set and cleared by software.
0: IO port H clock disabled during Sleep mode
1: IO port H clock enabled during Sleep mode

Bits 6 **GPIOGLPEN:** IO port G clock enable during Sleep mode
This bit is set and cleared by software.
0: IO port G clock disabled during Sleep mode
1: IO port G clock enabled during Sleep mode

Bit 5 **GPIOFLPEN:** IO port F clock enable during Sleep mode
This bit is set and cleared by software.
0: IO port F clock disabled during Sleep mode
1: IO port F clock enabled during Sleep mode

Bit 4 **GPIOELPEN:** IO port E clock enable during Sleep mode
Set and cleared by software.
0: IO port E clock disabled during Sleep mode
1: IO port E clock enabled during Sleep mode

Bit 3 **GPIODLPEN:** IO port D clock enable during Sleep mode
This bit is set and cleared by software.
0: IO port D clock disabled during Sleep mode
1: IO port D clock enabled during Sleep mode

Bit 2 **GPIOCLPEN:** IO port C clock enable during Sleep mode
This bit is set and cleared by software.
0: IO port C clock disabled during Sleep mode
1: IO port C clock enabled during Sleep mode

Bit 1 **GPIOBLPEN:** IO port B clock enable during Sleep mode
This bit is set and cleared by software.
0: IO port B clock disabled during Sleep mode
1: IO port B clock enabled during Sleep mode

Bit 0 **GPIOALPEN:** IO port A clock enable during sleep mode
This bit is set and cleared by software.
0: IO port A clock disabled during Sleep mode
1: IO port A clock enabled during Sleep mode

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

Address offset: 0x54

Reset value: 0x0000 00F1

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

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

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

Bit 7 **OTGFSLPEN:** USB OTG FS clock enable during Sleep mode
This bit is set and cleared by software.
0: USB OTG FS clock disabled during Sleep mode
1: USB OTG FS clock enabled during Sleep mode

Bit 6 **RNGLPEN:** Random number generator clock enable during Sleep mode
This bit is set and cleared by software.
0: Random number generator clock disabled during Sleep mode
1: Random number generator clock enabled during Sleep mode

Bit 5 **HASHLPEN:** Hash modules clock enable during Sleep mode
This bit is set and cleared by software.
0: Hash modules clock disabled during Sleep mode
1: Hash modules clock enabled during Sleep mode

Bit 4 **CRYPLPEN:** Cryptography modules clock enable during Sleep mode
This bit is set and cleared by software.
0: cryptography modules clock disabled during Sleep mode
1: cryptography modules clock enabled during Sleep mode

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

Bit 0 **DCMILPEN:** Camera interface enable during Sleep mode
This bit is set and cleared by software.
0: Camera interface clock disabled during Sleep mode
1: Camera interface clock enabled during Sleep mode

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

Address offset: 0x58

Reset value: 0x0000 0001

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

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

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

Bit 0 **FMCLPEN:** Flexible memory controller module clock enable during Sleep mode
This bit is set and cleared by software.
0: FMC module clock disabled during Sleep mode
1: FMC module clock enabled during Sleep mode

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

Address offset: 0x60

Reset value: 0xF6FE C9FF

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 LPEN	UART7 LPEN	DAC LPEN	PWR LPEN	RESERVED	CAN2 LPEN	CAN1 LPEN	Reser-ved	I2C3 LPEN	I2C2 LPEN	I2C1 LPEN	UART5 LPEN	UART4 LPEN	USART3 LPEN	USART2 LPEN	Reser-ved
rw	rw	rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 LPEN	SPI2 LPEN	Reserved		WWDG LPEN	Reserved		TIM14 LPEN	TIM13 LPEN	TIM12 LPEN	TIM7 LPEN	TIM6 LPEN	TIM5 LPEN	TIM4 LPEN	TIM3 LPEN	TIM2 LPEN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UART8LPEN:** UART8 clock enable during Sleep mode
This bit is set and cleared by software.
0: UART8 clock disabled during Sleep mode
1: UART8 clock enabled during Sleep mode

Bit 30 **UART7LPEN:** UART7 clock enable during Sleep mode
This bit is set and cleared by software.
0: UART7 clock disabled during Sleep mode
1: UART7 clock enabled during Sleep mode

Bit 29 **DACLPEN:** DAC interface clock enable during Sleep mode
This bit is set and cleared by software.
0: DAC interface clock disabled during Sleep mode
1: DAC interface clock enabled during Sleep mode

Bit 28 **PWRLPEN:** Power interface clock enable during Sleep mode
This bit is set and cleared by software.
0: Power interface clock disabled during Sleep mode
1: Power interface clock enabled during Sleep mode

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CAN2LPEN:** CAN 2 clock enable during Sleep mode
This bit is set and cleared by software.
0: CAN 2 clock disabled during sleep mode
1: CAN 2 clock enabled during sleep mode

Bit 25 **CAN1LPEN:** CAN 1 clock enable during Sleep mode
This bit is set and cleared by software.
0: CAN 1 clock disabled during Sleep mode
1: CAN 1 clock enabled during Sleep mode

Bit 24 Reserved, must be kept at reset value.

Bit 23 **I2C3LPEN:** I2C3 clock enable during Sleep mode
This bit is set and cleared by software.
0: I2C3 clock disabled during Sleep mode
1: I2C3 clock enabled during Sleep mode

Bit 22 **I2C2LPEN:** I2C2 clock enable during Sleep mode
This bit is set and cleared by software.
0: I2C2 clock disabled during Sleep mode
1: I2C2 clock enabled during Sleep mode

Bit 21 **I2C1LPEN:** I2C1 clock enable during Sleep mode
This bit is set and cleared by software.
0: I2C1 clock disabled during Sleep mode
1: I2C1 clock enabled during Sleep mode

Bit 20 **UART5LPEN:** UART5 clock enable during Sleep mode
This bit is set and cleared by software.
0: UART5 clock disabled during Sleep mode
1: UART5 clock enabled during Sleep mode

Bit 19 **UART4LPEN:** UART4 clock enable during Sleep mode
This bit is set and cleared by software.
0: UART4 clock disabled during Sleep mode
1: UART4 clock enabled during Sleep mode

Bit 18 **USART3LPEN:** USART3 clock enable during Sleep mode
This bit is set and cleared by software.
0: USART3 clock disabled during Sleep mode
1: USART3 clock enabled during Sleep mode

Bit 17 **USART2LPEN:** USART2 clock enable during Sleep mode
This bit is set and cleared by software.
0: USART2 clock disabled during Sleep mode
1: USART2 clock enabled during Sleep mode

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3LPEN:** SPI3 clock enable during Sleep mode
This bit is set and cleared by software.
0: SPI3 clock disabled during Sleep mode
1: SPI3 clock enabled during Sleep mode

Bit 14 **SPI2LPEN:** SPI2 clock enable during Sleep mode
This bit is set and cleared by software.
0: SPI2 clock disabled during Sleep mode
1: SPI2 clock enabled during Sleep mode

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

Bit 11 **WWDGLPEN:** Window watchdog clock enable during Sleep mode
This bit is set and cleared by software.
0: Window watchdog clock disabled during sleep mode
1: Window watchdog clock enabled during sleep mode

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

Bit 8 **TIM14LPEN:** TIM14 clock enable during Sleep mode
This bit is set and cleared by software.
0: TIM14 clock disabled during Sleep mode
1: TIM14 clock enabled during Sleep mode

Bit 7 **TIM13LPEN:** TIM13 clock enable during Sleep mode
This bit is set and cleared by software.
0: TIM13 clock disabled during Sleep mode
1: TIM13 clock enabled during Sleep mode

Bit 6 **TIM12LPEN:** TIM12 clock enable during Sleep mode
This bit is set and cleared by software.
0: TIM12 clock disabled during Sleep mode
1: TIM12 clock enabled during Sleep mode

Bit 5 **TIM7LPEN:** TIM7 clock enable during Sleep mode
This bit is set and cleared by software.
0: TIM7 clock disabled during Sleep mode
1: TIM7 clock enabled during Sleep mode

Bit 4 **TIM6LPEN:** TIM6 clock enable during Sleep mode
This bit is set and cleared by software.
0: TIM6 clock disabled during Sleep mode
1: TIM6 clock enabled during Sleep mode

Bit 3 **TIM5LPEN:** TIM5 clock enable during Sleep mode
This bit is set and cleared by software.
0: TIM5 clock disabled during Sleep mode
1: TIM5 clock enabled during Sleep mode

Bit 2 **TIM4LPEN:** TIM4 clock enable during Sleep mode
This bit is set and cleared by software.
0: TIM4 clock disabled during Sleep mode
1: TIM4 clock enabled during Sleep mode

Bit 1 **TIM3LPEN:** TIM3 clock enable during Sleep mode
This bit is set and cleared by software.
0: TIM3 clock disabled during Sleep mode
1: TIM3 clock enabled during Sleep mode

Bit 0 **TIM2LPEN:** TIM2 clock enable during Sleep mode
This bit is set and cleared by software.
0: TIM2 clock disabled during Sleep mode
1: TIM2 clock enabled during Sleep mode

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

Address offset: 0x64

Reset value: 0x0x0477 7F33

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					LTDC LPEN	Reserved			SAI1 LPEN	SPI6 LPEN	SPI5 LPEN	Reser-ved	TIM11 LPEN	TIM10 LPEN	TIM9 LPEN
					rw				rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG LPEN	SPI4 LPEN	SPI1 LPEN	SDIO LPEN	ADC3 LPEN	ADC2 LPEN	ADC1 LPEN	Reserved		USART 6 LPEN	USART 1 LPEN	Reserved		TIM8 LPEN	TIM1 LPEN
	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw

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

Bit 26 **LTDCLPEN:** LTDC clock enable during Sleep mode
This bit is set and cleared by software.
0: LTDC clock disabled during Sleep mode
1: LTDC clock enabled during Sleep mode

Bits 25:23 Reserved, must be kept at reset value.

Bit 22 **SAI1LPEN:** SAI1 clock enable during Sleep mode
This bit is set and cleared by software.
0: SAI1 clock disabled during Sleep mode
1: SAI1 clock enabled during Sleep mode

Bit 21 **SPI6LPEN:** SPI6 clock enable during Sleep mode
This bit is set and cleared by software.
0: SPI6 clock disabled during Sleep mode
1: SPI6 clock enabled during Sleep mode

Bit 20 **SPI5LPEN:** SPI5 clock enable during Sleep mode
This bit is set and cleared by software.
0: SPI5 clock disabled during Sleep mode
1: SPI5 clock enabled during Sleep mode

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TIM11LPEN:** TIM11 clock enable during Sleep mode
This bit is set and cleared by software.
0: TIM11 clock disabled during Sleep mode
1: TIM11 clock enabled during Sleep mode

Bit 17 **TIM10LPEN:** TIM10 clock enable during Sleep mode
This bit is set and cleared by software.
0: TIM10 clock disabled during Sleep mode
1: TIM10 clock enabled during Sleep mode

Bit 16 **TIM9LPEN:** TIM9 clock enable during sleep mode
This bit is set and cleared by software.
0: TIM9 clock disabled during Sleep mode
1: TIM9 clock enabled during Sleep mode

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SYSCFGLPEN:** System configuration controller clock enable during Sleep mode
This bit is set and cleared by software.
0: System configuration controller clock disabled during Sleep mode
1: System configuration controller clock enabled during Sleep mode

Bit 13 **SPI4LPEN:** SPI4 clock enable during Sleep mode
This bit is set and cleared by software.
0: SPI4 clock disabled during Sleep mode
1: SPI4 clock enabled during Sleep mode

Bit 12 **SPI1LPEN:** SPI1 clock enable during Sleep mode
This bit is set and cleared by software.
0: SPI1 clock disabled during Sleep mode
1: SPI1 clock enabled during Sleep mode

Bit 11 **SDIOLPEN:** SDIO clock enable during Sleep mode
This bit is set and cleared by software.
0: SDIO module clock disabled during Sleep mode
1: SDIO module clock enabled during Sleep mode

Bit 10 **ADC3LPEN:** ADC 3 clock enable during Sleep mode
This bit is set and cleared by software.
0: ADC 3 clock disabled during Sleep mode
1: ADC 3 clock disabled during Sleep mode

Bit 9 **ADC2LPEN:** ADC2 clock enable during Sleep mode
This bit is set and cleared by software.
0: ADC2 clock disabled during Sleep mode
1: ADC2 clock disabled during Sleep mode

Bit 8 **ADC1LPEN:** ADC1 clock enable during Sleep mode
This bit is set and cleared by software.
0: ADC1 clock disabled during Sleep mode
1: ADC1 clock disabled during Sleep mode

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

Bit 5 **USART6LPEN:** USART6 clock enable during Sleep mode
This bit is set and cleared by software.
0: USART6 clock disabled during Sleep mode
1: USART6 clock enabled during Sleep mode

Bit 4 **USART1LPEN:** USART1 clock enable during Sleep mode
This bit is set and cleared by software.
0: USART1 clock disabled during Sleep mode
1: USART1 clock enabled during Sleep mode

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

Bit 1 **TIM8LPEN:** TIM8 clock enable during Sleep mode
This bit is set and cleared by software.
0: TIM8 clock disabled during Sleep mode
1: TIM8 clock enabled during Sleep mode

Bit 0 **TIM1LPEN:** TIM1 clock enable during Sleep mode
This bit is set and cleared by software.
0: TIM1 clock disabled during Sleep mode
1: TIM1 clock enabled during Sleep mode

6.3.20 RCC Backup domain control register (RCC_BDCR)

Address offset: 0x70

Reset value: 0x0000 0000, reset by Backup domain reset.
Access: 0 ≤ wait state ≤ 3, word, half-word and byte access
Wait states are inserted in case of successive accesses to this register.

The LSEON, LSEBYP, RTCSEL and RTCEN bits in the *RCC Backup domain control register (RCC_BDCR)* are in the Backup domain. As a result, after Reset, these bits are write-protected and the DBP bit in the *PWR power control register (PWR_CR) for STM32F42xxx and STM32F43xxx* has to be set before these can be modified. Refer to *Section 6.1.1: System reset on page 150* for further information. These bits 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
Reserved															BDRST
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Reserved					RTCSEL[1:0]		Reserved					LSEBYP	LSERDY	LSEON
rw						rw	rw						rw	r	rw

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

Bit 16 **BDRST:** Backup domain software reset
This bit is set and cleared by software.
0: Reset not activated
1: Resets the entire Backup domain
Note: The BKPSRAM is not affected by this reset, the only way of resetting the BKPSRAM is through the Flash interface when a protection level change from level 1 to level 0 is requested.

Bit 15 **RTCEN:** RTC clock enable
This bit is 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

These bits are set by software to select the clock source for the RTC. Once the RTC clock source has been selected, it cannot be changed anymore unless the Backup domain is reset. The BDRST bit can be used to reset them.

00: No clock
01: LSE oscillator clock used as the RTC clock
10: LSI oscillator clock used as the RTC clock
11: HSE oscillator clock divided by a programmable prescaler (selection through the RTCPRE[4:0] bits in the RCC clock configuration register (RCC_CFGR)) used as the RTC clock

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

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

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

0: LSE oscillator not bypassed
1: LSE oscillator bypassed

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

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

0: LSE clock not ready
1: LSE clock ready

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

This bit is set and cleared by software.

0: LSE clock OFF
1: LSE clock ON

6.3.21 RCC clock control & status register (RCC_CSR)

Address offset: 0x74

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

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	BORRS TF	RMVF	Reserved							
r	r	r	r	r	r	r	rt_w								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														LSIRDY	LSION
														r	rw

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

This bit is set by hardware when a Low-power management reset occurs.
Cleared by writing to the RMVF bit.
0: No Low-power management reset occurred
1: Low-power management reset occurred
For further information on Low-power management reset, refer to *Low-power management reset*.

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

This bit is set by hardware when a window watchdog reset occurs.
Cleared by writing to the RMVF bit.
0: No window watchdog reset occurred
1: Window watchdog reset occurred

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

This bit is set by hardware when an independent watchdog reset from V_{DD} domain occurs.
Cleared by writing to the RMVF bit.
0: No watchdog reset occurred
1: Watchdog reset occurred

Bit 28 **SFTRSTF:** Software reset flag

This bit is set by hardware when a software reset occurs.
Cleared by writing to the RMVF bit.
0: No software reset occurred
1: Software reset occurred

Bit 27 **PORRSTF:** POR/PDR reset flag

This bit is set by hardware when a POR/PDR reset occurs.
Cleared by writing to the RMVF bit.
0: No POR/PDR reset occurred
1: POR/PDR reset occurred

Bit 26 **PINRSTF:** PIN reset flag

This bit is set by hardware when a reset from the NRST pin occurs.
Cleared by writing to the RMVF bit.
0: No reset from NRST pin occurred
1: Reset from NRST pin occurred

Bit 25 **BORRSTF:** BOR reset flag

Cleared by software by writing the RMVF bit.
This bit is set by hardware when a POR/PDR or BOR reset occurs.
0: No POR/PDR or BOR reset occurred
1: POR/PDR or BOR reset occurred

Bit 24 **RMVF:** Remove reset flag

This bit is set by software to clear the reset flags.
0: No effect
1: Clear the reset flags

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

Bit 1 **LSIRDY:** Internal low-speed oscillator ready

This bit is set and cleared by hardware to indicate when the internal RC 40 kHz oscillator is stable. After the LSION bit is cleared, LSIRDY goes low after 3 LSI clock cycles.

0: LSI RC oscillator not ready

1: LSI RC oscillator ready

Bit 0 **LSION:** Internal low-speed oscillator enable

This bit is set and cleared by software.

0: LSI RC oscillator OFF

1: LSI RC oscillator ON

6.3.22 RCC spread spectrum clock generation register (RCC_SSCGR)

Address offset: 0x80

Reset value: 0x0000 0000

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

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

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

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

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

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

This bit is set and cleared by software.

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

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

Bit 30 **SPREADSEL:** Spread Select

This bit is set and cleared by software.

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

0: Center spread

1: Down spread

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

Bits 27:13 **INCSTEP:** Incrementation step

These bits are set and cleared by software. To write before setting CR[24]=PLLON bit. Configuration input for modulation profile amplitude.

Bits 12:0 **MODPER:** Modulation period

These bits are set and cleared by software. To write before setting CR[24]=PLLON bit. Configuration input for modulation profile period.

6.3.23 RCC PLLI2S configuration register (RCC_PLLI2SCFGR)

Address offset: 0x84

Reset value: 0x2400 3000

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

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

$f_{(VCO\ clock)} = f_{(PLLI2S\ clock\ input)} \times (PLLI2SN / PLLM)$

$f_{(PLL\ I2S\ clock\ output)} = f_{(VCO\ clock)} / PLLI2SR$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	PLLI2SR2	PLLI2SR1	PLLI2SR0	PLLI2SQ				Reserved							
	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLLI2SN8	PLLI2SN7	PLLI2SN6	PLLI2SN5	PLLI2SN4	PLLI2SN3	PLLI2SN2	PLLI2SN1	PLLI2SN0	Reserved					
	rw	rw	rw	rw	rw	rw	rw	rw	rw						

Bit 31 Reserved, must be kept at reset value.

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

These bits are set and cleared by software to control the I2S clock frequency. These bits should be written only if the PLLI2S is disabled. The factor must be chosen in accordance with the prescaler values inside the I2S peripherals, to reach 0.3% error when using standard crystals and 0% error with audio crystals. For more information about I2S clock frequency and precision, refer to *Section 28.4.4: Clock generator* in the I2S chapter.

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

I2S clock frequency = VCO frequency / PLLR with 2 ≤PLLR ≤7
000: PLLR = 0, wrong configuration
001: PLLR = 1, wrong configuration
010: PLLR = 2
...
111: PLLR = 7

Bits 27:24 **PLLI2SQ**: PLLI2S division factor for SAI1 clock

These bits are set and cleared by software to control the SAI1 clock frequency. They should be written when the PLLI2S is disabled.
SAI1 clock frequency = VCO frequency / PLLI2SQ with 2 <= PLLI2SIQ <= 15
0000: PLLI2SQ = 0, wrong configuration
0001: PLLI2SQ = 1, wrong configuration
0010: PLLI2SQ = 2
0011: PLLI2SQ = 3
0100: PLLI2SQ = 4
0101: PLLI2SQ = 5
...
1111: PLLI2SQ = 15

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:6 **PLLI2SN:** PLLI2S multiplication factor for VCO

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

Caution: The software has to set these bits correctly to ensure that the VCO output frequency is between 100 and 432 MHz.

VCO output frequency = VCO input frequency × PLLI2SN with 50 ≤PLLI2SN ≤432

000000000: PLLI2SN = 0, wrong configuration

000000001: PLLI2SN = 1, wrong configuration

...

000110010: PLLI2SN = 50

...

001100011: PLLI2SN = 99

001100100: PLLI2SN = 100

001100101: PLLI2SN = 101

001100110: PLLI2SN = 102

...

110110000: PLLI2SN = 432

110110001: PLLI2SN = 433, wrong configuration

...

111111111: PLLI2SN = 511, wrong configuration

Note: Multiplication factors ranging from 50 and 99 are possible for VCO input frequency higher than 1 MHz. However care must be taken that the minimum VCO output frequency respects the value specified above.

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

6.3.24 RCC PLL configuration register (RCC_PLLSAICFGR)

Address offset: 0x88

Reset value: 0x2400 3000

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

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

- $f_{(VCO\ clock)} = f_{(PLLSAI\ clock\ input)} \times (PLLSAIN / PLLM)$
- $f_{(PLLSAI1\ clock\ output)} = f_{(VCO\ clock)} / PLLSAIQ$
- $f_{(PLL\ LCD\ clock\ output)} = f_{(VCO\ clock)} / PLLSAIR$

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

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **PLLSAIR:** PLLSAI division factor for LCD clock

Set and reset by software to control the LCD clock frequency.
These bits should be written when the PLLSAI is disabled.
LCD clock frequency = VCO frequency / PLLSAIR with 2 ≤PLLSAIR ≤7
000: PLLSAIR = 0, wrong configuration
001: PLLSAIR = 1, wrong configuration
010: PLLSAIR = 2
...
111: PLLSAIR = 7

Bits 27:24 **PLLSAIQ:** PLLSAI division factor for SAI1 clock

Set and reset by software to control the frequency of SAI1 clock.
These bits should be written when the PLLSAI is disabled.
SAI1 clock frequency = VCO frequency / PLLSAIQ with 2 ≤PLLSAIQ ≤15
0000: PLLSAIQ = 0, wrong configuration
0001: PLLSAIQ = 1, wrong configuration
...
0010: PLLSAIQ = 2
0011: PLLSAIQ = 3
0100: PLLSAIQ = 4
0101: PLLSAIQ = 5
...
1111: PLLSAIQ = 15

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:6 **PLLSAIN:** PLLSAI division factor for VCO

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

Caution: The software has to set these bits correctly to ensure that the VCO output frequency is between 100 and 432 MHz.

VCO output frequency = VCO input frequency × PLLISAIN with 50 ≤PLLISAIN ≤432

000000000: PLLISAIN = 0, wrong configuration

000000001: PLLISAIN = 1, wrong configuration

...

000110010: PLLISAIN = 50

...

001100011: PLLISAIN = 99

001100100: PLLISAIN = 100

001100101: PLLISAIN = 101

001100110: PLLISAIN = 102

...

110110000: PLLISAIN = 432

110110001: PLLISAIN = 433, wrong configuration

...

111111111: PLLISAIN = 511, wrong configuration

Note: Multiplication factors ranging from 50 and 99 are possible for VCO input frequency higher than 1 MHz. However care must be taken that the minimum VCO output frequency respects the value specified above.

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

6.3.25 RCC Dedicated Clock Configuration Register (RCC_DCKCFGR)

Address offset: 0x8C

Reset value: 0x0000 0000

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

The RCC_DCKCFGR register allows to configure the timer clock prescalers and the PLLSAI and PLLI2S output clock dividers for SAI1 and LTDC peripherals according to the following formula:

$f_{(PLLSAIDIVQ\ clock\ output)} = f_{(PLLSAI\_Q)} / PLLSAIDIVQ$

$f_{(PLLSAIDIVR\ clock\ output)} = f_{(PLLSAI\_R)} / PLLSAIDIVR$

$f_{(PLLI2SDIVQ\ clock\ output)} = f_{(PLLI2S\_Q)} / PLLI2SDIVQ$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							TIMPRE	SAI1BSRC		SAI1ASRC		Reserved		PLLSAIDIVR	
							rw	rw	rw	rw	rw			rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			PLLSAIDIVQ					Reserved			PLLS2DIVQ				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TIMPRE**: Timers clocks prescalers selection

This bit is set and reset by software to control the clock frequency of all the timers connected to APB1 and APB2 domain.

0: If the APB prescaler (PPRE1, PPRE2 in the RCC_CFGR register) is configured to a division factor of 1, TIMxCLK = PCLKx. Otherwise, the timer clock frequencies are set to twice to the frequency of the APB domain to which the timers are connected: TIMxCLK = 2xPCLKx.

1:If the APB prescaler (PPRE1, PPRE2 in the RCC_CFGR register) is configured to a division factor of 1, 2 or 4, TIMxCLK = HCLK. Otherwise, the timer clock frequencies are set to four times to the frequency of the APB domain to which the timers are connected: TIMxCLK = 4xPCLKx.

Bits 23:22 **SAI1BSRC**: SAI1-B clock source selection

These bits are set and cleared by software to control the SAI1-B clock frequency. They should be written when the PLLSAI and PLLI2S are disabled.

00: SAI1-B clock frequency = f(PLLSAI_Q) / PLLSAIDIVQ

01: SAI1-B clock frequency = f(PLLI2S_Q) / PLLI2SDIVQ

10: SAI1-B clock frequency = Alternate function input frequency

11: wrong configuration

Bits 21:20 **SAI1ASRC**: SAI1-A clock source selection

These bits are set and cleared by software to control the SAI1-A clock frequency. They should be written when the PLLSAI and PLLI2S are disabled.

00: SAI1-A clock frequency = f(PLLSAI_Q) / PLLSAIDIVQ

01: SAI1-A clock frequency = f(PLLI2S_Q) / PLLI2SDIVQ

10: SAI1-A clock frequency = Alternate function input frequency

11: wrong configuration

Bits 19: 18 Reserved, must be kept at reset value.

Bits 17:16 **PLLSAIDIVR**: division factor for LCD_CLK

These bits are set and cleared by software to control the frequency of LCD_CLK. They should be written only if PLLSAI is disabled.

LCD_CLK frequency = f(PLLSAI_R) / PLLSAIDIVR with 2 ≤PLLSAIDIVR ≤16

00: PLLSAIDIVR = /2

01: PLLSAIDIVR = /4

10: PLLSAIDIVR = /8

11: PLLSAIDIVR = /16

Bits 15: 13 Reserved, must be kept at reset value.

Bits 12:8 **PLLSAIDIVQ**: PLLSAI division factor for SAI1 clock

These bits are set and reset by software to control the SAI1 clock frequency.
They should be written only if PLLSAI is disabled.
SAI1 clock frequency = f(PLLSAI_Q) / PLLSAIDIVQ with 1 ≤PLLSAIDIVQ ≤31
00000: PLLSAIDIVQ = /1
00001: PLLSAIDIVQ = /2
00010: PLLSAIDIVQ = /3
00011: PLLSAIDIVQ = /4
00100: PLLSAIDIVQ = /5
...
11111: PLLSAIDIVQ = /32

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **PLLI2SDIVQ**: PLLI2S division factor for SAI1 clock

These bits are set and reset by software to control the SAI1 clock frequency.
They should be written only if PLLI2S is disabled.
SAI1 clock frequency = f(PLLI2S_Q) / PLLI2SDIVQ with 1 <= PLLI2SDIVQ <= 31
00000: PLLI2SDIVQ = /1
00001: PLLI2SDIVQ = /2
00010: PLLI2SDIVQ = /3
00011: PLLI2SDIVQ = /4
00100: PLLI2SDIVQ = /5
...
11111: PLLI2SDIVQ = /32

6.3.26 RCC register map

Table 33 gives the register map and reset values.

Table 33. RCC register map and reset values for STM32F42xxx and STM32F43xxx

Addr. 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	RCC_CR	Reserved		PLL SAIRDY	PLL SAION	PLL I2SRDY	PLL I2SON	PLL RDY	PLL ON	Reserved				CSSON	HSEBYP	HSERDY	HSEON	HSICAL 7	HSICAL 6	HSICAL 5	HSICAL 4	HSICAL 3	HSICAL 2	HSICAL 1	HSICAL 0	HSITRIM 4	HSITRIM 3	HSITRIM 2	HSITRIM 1	HSITRIM 0	Reserved	HSIRDY	HSION
0x04	RCC_PLLCFGR	Reserved				PLLQ 3	PLLQ 2	PLLQ 1	PLLQ 0	Reserved	PLLSRC	Reserved				PLLP 1	PLLP 0	Reserved	PLLN 8	PLLN 7	PLLN 6	PLLN 5	PLLN 4	PLLN 3	PLLN 2	PLLN 1	PLLN 0	PLLM 5	PLLM 4	PLLM 3	PLLM 2	PLLM 1	PLLM 0
0x08	RCC_CFGR	MCO2 1	MCO2 0	MCO2PRE2	MCO2PRE1	MCO2PRE0	MCO1PRE2	MCO1PRE1	MCO1PRE0	I2SSRC	MCO1 1	MCO1 0	RTCPRE 4	RTCPRE 3	RTCPRE 2	RTCPRE 1	RTCPRE 0	PPRE2 2	PPRE2 1	PPRE2 0	PPRE1 2	PPRE1 1	PPRET 0	Reserved		HPRE 3	HPRE 2	HPRE 1	HPRE 0	SWS 1	SWS 0	SW 1	SW 0
0x0C	RCC_CIR	Reserved								CSSC	PLLSAIRDYC	PLLI2SRDYC	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	Reserved	PLLSAIRDYIE	PLLI2SRDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	PLLSAIRDYF	PLLI2SRDYF	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
0x10	RCC_AHB1RSTR	Reserved		OTGHSRST	Reserved			ETHMACRST	Reserved	DMA2DRST	DMA2RST	DMA1RST	Reserved								CRCRST	Reserved	GPIOKRST	GPIOJRST	GPIOIRST	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
0x14	RCC_AHB2RSTR	Reserved																								OTGFSRST	RNGRST	HSAHRST	CRYPRST	Reserved			DCMIRST
0x18	RCC_AHB3RSTR	Reserved																															FMCRST
0x1C	Reserved	Reserved																															
0x20	RCC_APB1RSTR	UART8RST	UART7RST	DACRST	PWRRST	Reserved	CAN2RST	CAN1RST	Reserved	I2C3RST	I2C2RST	I2C1RST	UART5RST	UART4RST	UART3RST	UART2RST	Reserved	SPI3RST	SPI2RST	Reserved		WWDGRST	Reserved		TIM14RST	TIM13RST	TIM12RST		TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
0x24	RCC_APB2RSTR	Reserved					LTDCRST	Reserved		SAI1RST	SPI6RST	SPI5RST	Reserved	TIM11RST	TIM10RST	TIM9RST	Reserved	SYSCFGRST	SP45RST	SPI1RST	SDIORST	Reserved		ADCRST	Reserved		USART6RST	USART1RST	Reserved		TIM8RST	TIM1RST
0x28	Reserved	Reserved																															
0x2C	Reserved	Reserved																															
0x30	RCC_AHB1ENR	Reserved	OTGHSULPIEN	OTGHSEN	ETHMACPTPEN	ETHMACRXEN	ETHMACTXEN	ETHMACEN	Reserved	DMA2DEN	DMA2EN	DMA1EN	CCMDATARAMEN	Reserved	BKPSRAMEN	Reserved					CRCEN	Reserved	GPIOKEN	GPIOJEN	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
0x34	RCC_AHB2ENR	Reserved																								OTGFSEN	RNGEN	HASHEN	CRYPEN	Reserved			DCMIEN
0x38	RCC_AHB3ENR	Reserved																															FMCEN
0x3C	Reserved	Reserved																															

Table 33. RCC register map and reset values for STM32F42xxx and STM32F43xxx (continued)

Addr. 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
0x40	RCC_ APB1ENR	UART8EN	UART7EN	DACEN	PWREN	Reserved	CAN2EN	CAN1EN	Reserved	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Reserved	SPI3EN	SPI2EN	Reserved		WWDGEN	Reserved		TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
0x44	RCC_ APB2ENR	Reserved					LTDCEN	Reserved		SAI1EN	SPI6EN	SPI5EN	Reserved	TIM11EN	TIM10EN	TIM9EN	Reserved	SYSCFGEN	SPI4EN	SPI1EN	SDIOEN	ADC3EN	ADC2EN	ADC1EN	Reserved	USART6EN	USART1EN	Reserved		TIM8EN	TIM1EN		
0x48	Reserved	Reserved																															
0x4C	Reserved	Reserved																															
0x50	RCC_AHB1LP ENR	Reserved	OTGHSULPILPEN	OTGHSLPEN	ETHMACPTPLPEN	ETHMACRXLPEN	ETHMACTXLPEN	ETHMACLPEN	Reserved	DMA2DLPEN	DMA2LPEN	DMA1LPEN	Reserved	SRAM3LPEN	BKPSRAMLPEN	SRAM2LPEN	SRAM1LPEN	FLITFLPEN	Reserved		CRCLPEN	Reserved	GPIOKLPEN	GPIOJLPEN	GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
0x54	RCC_AHB2LP ENR	Reserved																								OTGFSLPEN	RNGLPEN	HASHLPEN	CRYPLPEN	Reserved			DCMILPEN
0x58	RCC_AHB3LP ENR	Reserved																															FMCLPEN
0x5C	Reserved	Reserved																															
0x60	RCC_APB1LP ENR	UART8LPEN	UART7LPEN	DACLPEN	PWRLPEN	Reserved	CAN2LPEN	CAN1LPEN	Reserved	I2C3LPEN	I2C2LPEN	I2C1LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	Reserved	SPI3LPEN	SPI2LPEN	Reserved		WWDGLPEN	Reserved		TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
0x64	RCC_APB2LP ENR	Reserved					LTDCLPEN	Reserved		SAI1LPEN	SPI6LPEN	SPI5LPEN	Reserved	TIM11LPEN	TIM10LPEN	TIM9LPEN	Reserved	SYSCFGLPEN	SPI4LPEN	SPI1LPEN	SDIOLPEN	ADC3LPEN	ADC2LPEN	ADC1LPEN	Reserved	USART6LPEN	USART1LPEN	Reserved		TIM8LPEN	TIM1LPEN		
0x68	Reserved	Reserved																															
0x6C	Reserved	Reserved																															
0x70	RCC_BDCR	Reserved															BDRST	RTCEN	Reserved				RTCSEL 1	RTCSEL 0	Reserved					LSEBYP	LSERDY	LSEON
0x74	RCC_CSR	LPWRRSTF	WWDGRSTF	WDGRSTF	SFTRSTF	PORRSTF	PADRSTF	BORRSTF	RMVF	Reserved																					LSIRDY	LSION
0x78	Reserved	Reserved																															
0x7C	Reserved	Reserved																															
0x80	RCC_SSCGR	SSCGEN	SPREADSEL	Reserved			INCSTEP														MODPER												
0x84	RCC_PLLI2SC FGR	Reserved	PLLI2SRx				PLLI2SQ			Reserved									PLLI2SNx						Reserved								

Table 33. RCC register map and reset values for STM32F42xxx and STM32F43xxx (continued)

Addr. 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
0x88	RCC_PLLSAICFGR		PLLSAIR				PLLSAIQ			Reserved									PLLSAIN									Reserved					
0x8C	RCC_DCKCFGR	Reserved							TIMPRE	SAI1BSCR		SAI1ASCR		Reserved		PLLSAIDIVR		Reserved	PLLSAIDIVQ					Reserved				PLLI2SDIVQ				

Refer to *Section 2.3: Memory map* for the register boundary addresses.

7 Reset and clock control for STM32F405xx/07xx and STM32F415xx/17xx(RCC)

7.1 Reset

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

7.1.1 System reset

A system reset sets all registers to their reset values except the reset flags in the clock controller CSR register and the registers in the Backup domain (see *Figure 20*).

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

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

Software reset

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

The SYSRESETREQ bit in Cortex®-M4 with FPU Application Interrupt and Reset Control Register must be set to force a software reset on the device. Refer to the Cortex®-M4 with FPU technical reference manual for more details.

Low-power management reset

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

1. Reset generated when entering the Standby mode:

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

2. Reset when entering the Stop mode:

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

7.1.2 Power reset

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

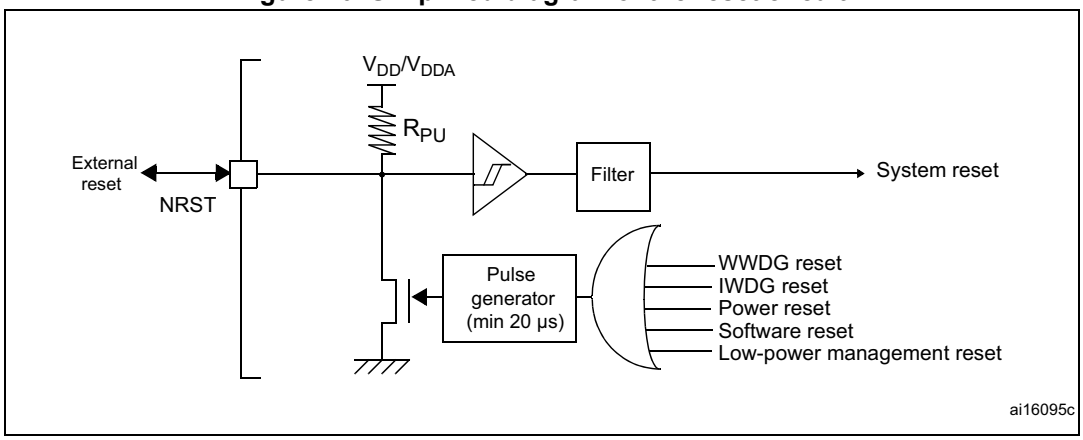
1. Power-on/power-down reset (POR/PDR reset) or brownout (BOR) reset
2. When exiting the Standby mode

A power reset sets all registers to their reset values except the Backup domain (see *Figure 20*)

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

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

Figure 20. Simplified diagram of the reset circuit



The Backup domain has two specific resets that affect only the Backup domain (see *Figure 20*).

7.1.3 Backup domain reset

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

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

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

7.2 Clocks

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

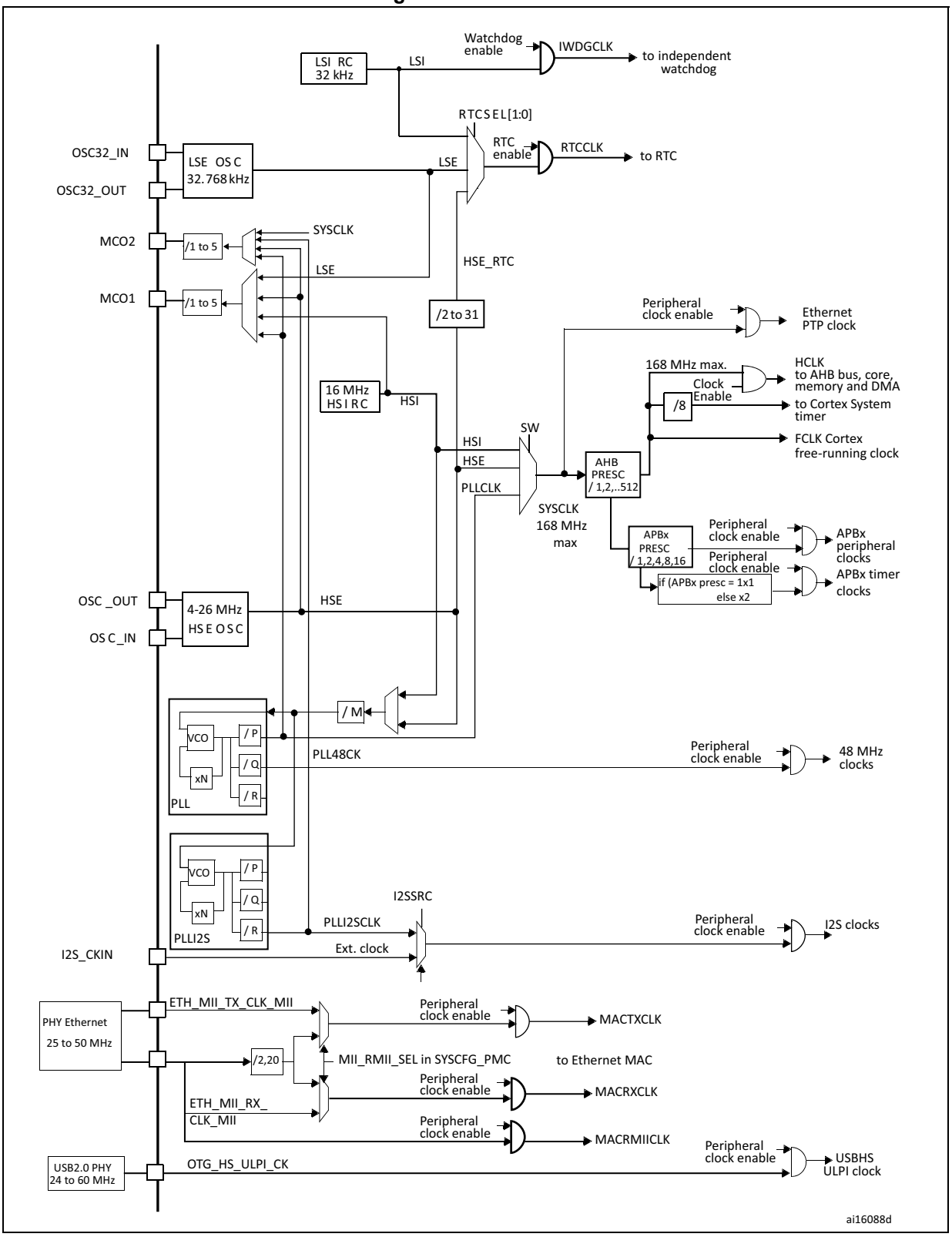
- HSI oscillator clock
- HSE oscillator clock
- Main PLL (PLL) clock

The devices have the two following secondary clock sources:

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

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

Figure 21. Clock tree



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

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

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

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

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

  To achieve high-quality audio performance, the I2S clock can be derived either from a specific PLL (PLLI2S) or from an external clock mapped on the I2S_CKIN pin. For more information about I2S clock frequency and precision, refer to *Section 28.4.4: Clock generator*.
- The USB OTG HS (60 MHz) clock which is provided from the external PHY
- The Ethernet MAC clocks (TX, RX and RMII) which are provided from the external PHY. For further information on the Ethernet configuration, please refer to *Section 33.4.4: MII/RMII selection* in the Ethernet peripheral description. When the Ethernet is used, the AHB clock frequency must be at least 25 MHz.

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

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

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

FCLK acts as Cortex®-M4 with FPU free-running clock. For more details, refer to the Cortex®-M4 with FPU technical reference manual.

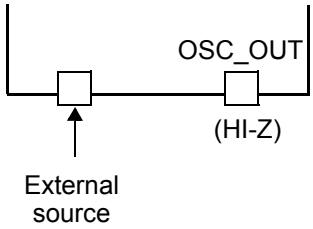
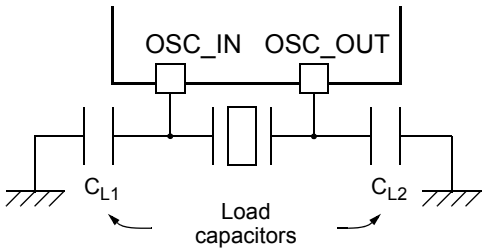
7.2.1 HSE clock

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

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

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

Figure 22. HSE/ LSE clock sources

	Hardware configuration
External clock	
Crystal/ceramic resonators	

External source (HSE bypass)

In this mode, an external clock source must be provided. You select this mode by setting the HSEBYP and HSEON bits in the *RCC clock control register (RCC_CR)*. The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC_IN pin while the OSC_OUT pin should be left HI-Z. See *Figure 22*.

External crystal/ceramic resonator (HSE crystal)

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

The associated hardware configuration is shown in *Figure 22*. Refer to the electrical characteristics section of the *datasheet* for more details.

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

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

7.2.2 HSI clock

The HSI clock signal is generated from an internal 16 MHz RC oscillator and can be used directly as a system clock, or used as PLL input.

The HSI RC oscillator has the advantage of providing a clock source at low cost (no external components). It also has a faster startup time than the HSE crystal oscillator however, even with calibration the frequency is less accurate than an external crystal oscillator or ceramic resonator.

Calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by ST for 1% accuracy at T_A= 25 °C.

After reset, the factory calibration value is loaded in the HSICAL[7:0] bits in the *RCC clock control register (RCC_CR)*.

If the application is subject to voltage or temperature variations this may affect the RC oscillator speed. You can trim the HSI frequency in the application using the HSITRIM[4:0] bits in the *RCC clock control register (RCC_CR)*.

The HSIRDY flag in the *RCC clock control register (RCC_CR)* indicates if the HSI RC is stable or not. At startup, the HSI RC output clock is not released until this bit is set by hardware.

The HSI RC can be switched on and off using the HSION bit in the *RCC clock control register (RCC_CR)*.

The HSI signal can also be used as a backup source (Auxiliary clock) if the HSE crystal oscillator fails. Refer to *Section 7.2.7: Clock security system (CSS) on page 220*.

7.2.3 PLL configuration

The STM32F4xx devices feature two PLLs:

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

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

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

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

7.2.4 LSE clock

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

The LSE oscillator is switched on and off using the LSEON bit in *RCC Backup domain control register (RCC_BDCR)*.

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

External source (LSE bypass)

In this mode, an external clock source must be provided. It must have a frequency up to 1 MHz. You select this mode by setting the LSEBYP and LSEON bits in the *RCC Backup domain control register (RCC_BDCR)*. The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC32_IN pin while the OSC32_OUT pin should be left HI-Z. See *Figure 22*.

7.2.5 LSI clock

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

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

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

7.2.6 System clock (SYSCLK) selection

After a system reset, the HSI oscillator is selected as the system clock. When a clock source is used directly or through PLL as the system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source that is not yet ready is selected, the switch occurs when the clock source is ready. Status bits in the *RCC clock control register (RCC_CR)* indicate which clock(s) is (are) ready and which clock is currently used as the system clock.

7.2.7 Clock security system (CSS)

The clock security system can be activated by software. In this case, the clock detector is enabled after the HSE oscillator startup delay, and disabled when this oscillator is stopped.

If a failure is detected on the HSE clock, this oscillator is automatically disabled, a clock failure event is sent to the break inputs of advanced-control timers TIM1 and TIM8, and an interrupt is generated to inform the software about the failure (clock security system interrupt CSSI), allowing the MCU to perform rescue operations. The CSSI is linked to the Cortex®-M4 with FPU NMI (non-maskable interrupt) exception vector.

Note: When the CSS is enabled, if the HSE clock happens to fail, the CSS generates an interrupt, which causes the automatic generation of an NMI. The NMI is executed indefinitely unless the CSS interrupt pending bit is cleared. As a consequence, the application has to clear the CSS interrupt in the NMI ISR by setting the CSSC bit in the Clock interrupt register (RCC_CIR).

If the HSE oscillator is used directly or indirectly as the system clock (indirectly meaning that it is directly used as PLL input clock, and that PLL clock is the system clock) and a failure is detected, then the system clock switches to the HSI oscillator and the HSE oscillator is disabled.

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

7.2.8 RTC/AWU clock

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

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

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

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

- If LSE is selected as the RTC clock:
  - The RTC continues to work even if the V_{DD} supply is switched off, provided the V_{BAT} supply is maintained.
- If LSI is selected as the Auto-wakeup unit (AWU) clock:
  - The AWU state is not guaranteed if the V_{DD} supply is powered off. Refer to *Section 7.2.5: LSI clock on page 220* for more details on LSI calibration.
- If the HSE clock is used as the RTC clock:
  - The RTC state is not guaranteed if the V_{DD} supply is powered off or if the internal voltage regulator is powered off (removing power from the 1.2 V domain).

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

7.2.9 Watchdog clock

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

7.2.10 Clock-out capability

Two microcontroller clock output (MCO) pins are available:

- MCO1

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

  – HSI clock
  – LSE clock
  – HSE clock
  – PLL clock

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

- MCO2

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

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

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

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

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

7.2.11 Internal/external clock measurement using TIM5/TIM11

It is possible to indirectly measure the frequencies of all on-board clock source generators by means of the input capture of TIM5 channel4 and TIM11 channel1 as shown in *Figure 23* and *Figure 24*.

Internal/external clock measurement using TIM5 channel4

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

The primary purpose of having the LSE connected to the channel4 input capture is to be able to precisely measure the HSI (this requires to have the HSI used as the system clock source). The number of HSI clock counts between consecutive edges of the LSE signal provides a measurement of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppm) we can determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process and/or temperature- and voltage-related frequency deviations.

The HSI oscillator has dedicated, user-accessible calibration bits for this purpose.

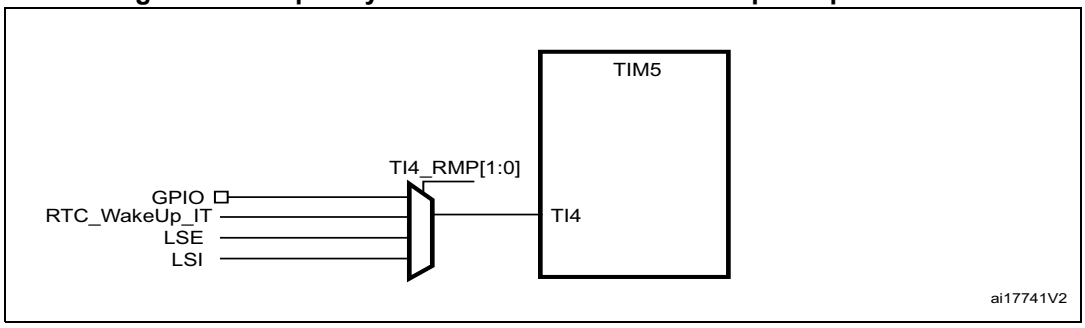
The basic concept consists in providing a relative measurement (e.g. HSI/LSE ratio): the precision is therefore tightly linked to the ratio between the two clock sources. The greater the ratio, the better the measurement.

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

Use the following procedure to measure the LSI frequency:

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

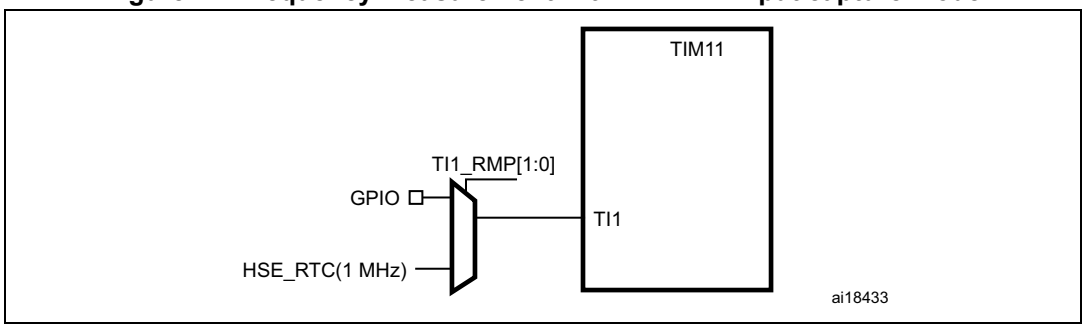
Figure 23. Frequency measurement with TIM5 in Input capture mode



Internal/external clock measurement using TIM11 channel1

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

Figure 24. Frequency measurement with TIM11 in Input capture mode



7.3 RCC registers

Refer to *Section 1.1: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

7.3.1 RCC clock control register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 XX83 where X is undefined.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				PLLI2SRDY	PLLI2SON	PLLRDY	PLLON	Reserved				CSSON	HSEBYP	HSERDY	HSEON
				r	rw	r	rw					rw	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res.	HSIRDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

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

Bit 27 **PLLI2SRDY**: PLLI2S clock ready flag
Set by hardware to indicate that the PLLI2S is locked.
0: PLLI2S unlocked
1: PLLI2S locked

Bit 26 **PLLI2SON**: PLLI2S enable
Set and cleared by software to enable PLLI2S.
Cleared by hardware when entering Stop or Standby mode.
0: PLLI2S OFF
1: PLLI2S ON

Bit 25 **PLLRDY**: Main PLL (PLL) clock ready flag
Set by hardware to indicate that PLL is locked.
0: PLL unlocked
1: PLL locked

Bit 24 **PLLON**: Main PLL (PLL) enable
Set and cleared by software to enable PLL.
Cleared by hardware when entering Stop or Standby mode. This bit cannot be reset if PLL clock is used as the system clock.
0: PLL OFF
1: PLL ON

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **CSSON**: Clock security system enable
Set and cleared by software to enable the clock security system. When CSSON is set, the clock detector is enabled by hardware when the HSE oscillator is ready, and disabled by hardware if an oscillator failure is detected.
0: Clock security system OFF (Clock detector OFF)
1: Clock security system ON (Clock detector ON if HSE oscillator is stable, OFF if not)

Bit 18 **HSEBYP**: HSE clock bypass

Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit, to be used by the device.
The HSEBYP bit can be written only if the HSE oscillator is disabled.
0: HSE oscillator not bypassed
1: HSE oscillator bypassed with an external clock

Bit 17 **HSERDY**: HSE clock ready flag

Set by hardware to indicate that the HSE oscillator is stable. After the HSEON bit is cleared, HSERDY goes low after 6 HSE oscillator clock cycles.
0: HSE oscillator not ready
1: HSE oscillator ready

Bit 16 **HSEON**: HSE clock enable

Set and cleared by software.
Cleared by hardware to stop the HSE oscillator when entering Stop or Standby mode. This bit cannot be reset if the HSE oscillator is used directly or indirectly as the system clock.
0: HSE oscillator OFF
1: HSE oscillator ON

Bits 15:8 **HSICAL[7:0]**: Internal high-speed clock calibration

These bits are initialized automatically at startup.

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

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

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HSIRDY**: Internal high-speed clock ready flag

Set by hardware to indicate that the HSI oscillator is stable. After the HSION bit is cleared, HSIRDY goes low after 6 HSI clock cycles.
0: HSI oscillator not ready
1: HSI oscillator ready

Bit 0 **HSION**: Internal high-speed clock enable

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

7.3.2 RCC PLL configuration register (RCC_PLLCFGR)

Address offset: 0x04

Reset value: 0x2400 3010

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

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				PLLQ3	PLLQ2	PLLQ1	PLLQ0	Reserved	PLLSRC	Reserved				PLLP1	PLLP0
				rw	rw	rw	rw		rw					rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLLN									PLLM5	PLLM4	PLLM3	PLLM2	PLLM1	PLLM0
	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 27:24 **PLLQ:** Main PLL (PLL) division factor for USB OTG FS, SDIO and random number generator clocks

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

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

USB OTG FS clock frequency = VCO frequency / PLLQ with 2 ≤PLLQ ≤15

0000: PLLQ = 0, wrong configuration

0001: PLLQ = 1, wrong configuration

0010: PLLQ = 2

0011: PLLQ = 3

0100: PLLQ = 4

...

1111: PLLQ = 15

Bit 23 Reserved, must be kept at reset value.

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

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

0: HSI clock selected as PLL and PLLI2S clock entry

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

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

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

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

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

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

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

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

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

Caution: The software has to set these bits correctly to ensure that the VCO output frequency is between 100 and 432 MHz.

VCO output frequency = VCO input frequency × PLLN with 50 ≤PLLN ≤432

000000000: PLLN = 0, wrong configuration
000000001: PLLN = 1, wrong configuration
...
000110010: PLLN = 50
...
001100011: PLLN = 99
001100100: PLLN = 100
...
110110000: PLLN = 432
110110001: PLLN = 433, wrong configuration
...
111111111: PLLN = 511, wrong configuration

Note: Multiplication factors ranging from 50 and 99 are possible for VCO input frequency higher than 1 MHz. However care must be taken that the minimum VCO output frequency respects the value specified above.

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

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

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

VCO input frequency = PLL input clock frequency / PLLM with 2 ≤PLLM ≤63

000000: PLLM = 0, wrong configuration
000001: PLLM = 1, wrong configuration
000010: PLLM = 2
000011: PLLM = 3
000100: PLLM = 4
...
111110: PLLM = 62
111111: PLLM = 63

7.3.3 RCC clock configuration register (RCC_CFGR)

Address offset: 0x08

Reset value: 0x0000 0000

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2		MCO2 PRE[2:0]			MCO1 PRE[2:0]			I2SSCR	MCO1		RTCPRE[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
PPRE2[2:0]			PPRE1[2:0]			Reserved		HPRE[3:0]				SWS1	SWS0	SW1	SW0
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	r	r	rw	rw

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

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

00: System clock (SYSCLK) selected

01: PLLI2S clock selected

10: HSE oscillator clock selected

11: PLL clock selected

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

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

0xx: no division

100: division by 2

101: division by 3

110: division by 4

111: division by 5

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

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

0xx: no division

100: division by 2

101: division by 3

110: division by 4

111: division by 5

Bit 23 **I2SSRC**: I2S clock selection

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

0: PLLI2S clock used as I2S clock source

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

Bits 22:21 **MCO1:** Microcontroller clock output 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

0xxx: system clock not divided
1000: system clock divided by 2
1001: system clock divided by 4
1010: system clock divided by 8
1011: system clock divided by 16
1100: system clock divided by 64
1101: system clock divided by 128
1110: system clock divided by 256
1111: system clock divided by 512

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

Set and cleared by hardware to indicate which clock source is used as the system clock.
00: HSI oscillator used as the system clock
01: HSE oscillator used as the system clock
10: PLL used as the system clock
11: not applicable

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

Set and cleared by software to select the system clock source.
Set by hardware to force the HSI selection when leaving the Stop or Standby mode or in case of failure of the HSE oscillator used directly or indirectly as the system clock.
00: HSI oscillator selected as system clock
01: HSE oscillator selected as system clock
10: PLL selected as system clock
11: not allowed

7.3.4 RCC clock interrupt register (RCC_CIR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CSSC	Reserved	PLLI2S RDYC	PLL RDYC	HSE RDYC	HSI RDYC	LSE RDYC	LSI RDYC
								w		w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		PLLI2S RDYIE	PLL RDYIE	HSE RDYIE	HSI RDYIE	LSE RDYIE	LSI RDYIE	CSSF	Reserved	PLLI2S RDYF	PLL RDYF	HSE RDYF	HSI RDYF	LSE RDYF	LSI RDYF
		rw	rw	rw	rw	rw	rw	r		r	r	r	r	r	r

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

Bit 23 **CSSC:** Clock security system interrupt clear
This bit is set by software to clear the CSSF flag.
0: No effect
1: Clear CSSF flag

Bits 22 Reserved, must be kept at reset value.

Bit 21 **PLLI2SRDYC:** PLLI2S ready interrupt clear
This bit is set by software to clear the PLLI2SRDYF flag.
0: No effect
1: PLLI2SRDYF cleared

Bit 20 **PLLRDYC:** Main PLL(PLL) ready interrupt clear
This bit is set by software to clear the PLLRDYF flag.
0: No effect
1: PLLRDYF cleared

Bit 19 **HSERDYC:** HSE ready interrupt clear
This bit is set by software to clear the HSERDYF flag.
0: No effect
1: HSERDYF cleared

Bit 18 **HSIRDYC:** HSI ready interrupt clear
This bit is set software to clear the HSIRDYF flag.
0: No effect
1: HSIRDYF cleared

Bit 17 **LSERDYC:** LSE ready interrupt clear
This bit is set by software to clear the LSERDYF flag.
0: No effect
1: LSERDYF cleared

Bit 16 **LSIRDYC:** LSI ready interrupt clear
This bit is set by software to clear the LSIRDYF flag.
0: No effect
1: LSIRDYF cleared

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

Bit 13 **PLLI2SRDYIE:** PLLI2S ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLLI2S lock.
0: PLLI2S lock interrupt disabled
1: PLLI2S lock interrupt enabled

Bit 12 **PLLRDYIE:** Main PLL (PLL) ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL lock.
0: PLL lock interrupt disabled
1: PLL lock interrupt enabled

Bit 11 **HSERDYIE:** HSE ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the HSE oscillator stabilization.
0: HSE ready interrupt disabled
1: HSE ready interrupt enabled

Bit 10 **HSIRDYIE:** HSI ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the HSI oscillator stabilization.
0: HSI ready interrupt disabled
1: HSI ready interrupt enabled

Bit 9 **LSERDYIE:** LSE ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the LSE oscillator stabilization.
0: LSE ready interrupt disabled
1: LSE ready interrupt enabled

Bit 8 **LSIRDYIE:** LSI ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by LSI oscillator stabilization.
0: LSI ready interrupt disabled
1: LSI ready interrupt enabled

Bit 7 **CSSF:** Clock security system interrupt flag
Set by hardware when a failure is detected in the HSE oscillator.
Cleared by software setting the CSSC bit.
0: No clock security interrupt caused by HSE clock failure
1: Clock security interrupt caused by HSE clock failure

Bits 6 Reserved, must be kept at reset value.

Bit 5 **PLLI2SRDYF:** PLLI2S ready interrupt flag
Set by hardware when the PLLI2S locks and PLLI2SRDYDIE is set.
Cleared by software setting the PLLRI2SDYC bit.
0: No clock ready interrupt caused by PLLI2S lock
1: Clock ready interrupt caused by PLLI2S lock

Bit 4 **PLLRDYF:** Main PLL (PLL) ready interrupt flag
Set by hardware when PLL locks and PLLRDYDIE is set.
Cleared by software setting the PLLRDYC bit.
0: No clock ready interrupt caused by PLL lock
1: Clock ready interrupt caused by PLL lock

Bit 3 **HSERDYF:** HSE ready interrupt flag
Set by hardware when External High Speed clock becomes stable and HSERDYDIE is set.
Cleared by software setting the HSERDYC bit.
0: No clock ready interrupt caused by the HSE oscillator
1: Clock ready interrupt caused by the HSE oscillator

Bit 2 **HSIRDYF:** HSI ready interrupt flag
Set by hardware when the Internal High Speed clock becomes stable and HSIRDYDIE is set.
Cleared by software setting the HSIRDYC bit.
0: No clock ready interrupt caused by the HSI oscillator
1: Clock ready interrupt caused by the HSI oscillator

Bit 1 **LSERDYF:** LSE ready interrupt flag
Set by hardware when the External Low Speed 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 internal low speed clock becomes stable and LSIRDYDIE is set.
Cleared by software setting the LSIRDYC bit.
0: No clock ready interrupt caused by the LSI oscillator
1: Clock ready interrupt caused by the LSI oscillator

7.3.5 RCC AHB1 peripheral reset register (RCC_AHB1RSTR)

Address offset: 0x10

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		OTGHS RST	Reserved			ETHMAC RST	Reserved		DMA2 RST	DMA1 RST	Reserved				
		rw				rw			rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CRCRST	Reserved			GPIOIRST	GPIOHRST	GPIOGGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 29 **OTGHSRST:** USB OTG HS module reset
Set and cleared by software.
0: does not reset the USB OTG HS module
1: resets the USB OTG HS module

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

Bit 25 **ETHMACRST:** Ethernet MAC reset
Set and cleared by software.
0: does not reset Ethernet MAC
1: resets Ethernet MAC

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

Bit 22 **DMA2RST:** DMA2 reset
Set and cleared by software.
0: does not reset DMA2
1: resets DMA2

Bit 21 **DMA1RST:** DMA1 reset
Set and cleared by software.
0: does not reset DMA1
1: resets DMA1

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

Bit 12 **CRCRST:** CRC reset
Set and cleared by software.
0: does not reset CRC
1: resets CRC

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

Bit 8 **GPIOIRST:** IO port I reset
Set and cleared by software.
0: does not reset IO port I
1: resets IO port I

Bit 7 **GPIOHRST:** IO port H reset
Set and cleared by software.
0: does not reset IO port H
1: resets IO port H

Bits 6 **GPIOGRST:** IO port G reset
Set and cleared by software.
0: does not reset IO port G
1: resets IO port G

Bit 5 **GPIOFRST:** IO port F reset
Set and cleared by software.
0: does not reset IO port F
1: resets IO port F

Bit 4 **GPIOERST:** IO port E reset
Set and cleared by software.
0: does not reset IO port E
1: resets IO port E

Bit 3 **GPIODRST:** IO port D reset
Set and cleared by software.
0: does not reset IO port D
1: resets IO port D

Bit 2 **GPIOCRST:** IO port C reset
Set and cleared by software.
0: does not reset IO port C
1: resets IO port C

Bit 1 **GPIOBRST:** IO port B reset
Set and cleared by software.
0: does not reset IO port B
1:resets IO port B

Bit 0 **GPIOARST:** IO port A reset
Set and cleared by software.
0: does not reset IO port A
1: resets IO port A

7.3.6 RCC AHB2 peripheral reset register (RCC_AHB2RSTR)

Address offset: 0x14

Reset value: 0x0000 0000

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

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

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

Bit 7 **OTGFSRST:** USB OTG FS module reset
Set and cleared by software.
0: does not reset the USB OTG FS module
1: resets the USB OTG FS module

Bit 6 **RNGRST:** Random number generator module reset
Set and cleared by software.
0: does not reset the random number generator module
1: resets the random number generator module

Bit 5 **HASHRST:** Hash module reset
Set and cleared by software.
0: does not reset the HASH module
1: resets the HASH module

Bit 4 **CRYPRST:** Cryptographic module reset
Set and cleared by software.
0: does not reset the cryptographic module
1: resets the cryptographic module

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

Bit 0 **DCMIRST:** Camera interface reset
Set and cleared by software.
0: does not reset the Camera interface
1: resets the Camera interface

7.3.7 RCC AHB3 peripheral reset register (RCC_AHB3RSTR)

Address offset: 0x18

Reset value: 0x0000 0000

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

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

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

Bit 0 **FSMCRST:** Flexible static memory controller module reset
Set and cleared by software.
0: does not reset the FSMC module
1: resets the FSMC module

7.3.8 RCC APB1 peripheral reset register (RCC_APB1RSTR)

Address offset: 0x20

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DACRST	PWR RST	Reser-ved	CAN2 RST	CAN1 RST	Reser-ved	I2C3 RST	I2C2 RST	I2C1 RST	UART5 RST	UART4 RST	UART3 RST	UART2 RST	Reser-ved
		rw	rw			rw		rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 RST	SPI2 RST	Reserved		WWDG RST	Reserved		TIM14 RST	TIM13 RST	TIM12 RST	TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST	TIM2 RST
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 29 **DACRST:** DAC reset
Set and cleared by software.
0: does not reset the DAC interface
1: resets the DAC interface

Bit 28 **PWRRST:** Power interface reset
Set and cleared by software.
0: does not reset the power interface
1: resets the power interface

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CAN2RST:** CAN2 reset
Set and cleared by software.
0: does not reset CAN2
1: resets CAN2

Bit 25 **CAN1RST:** CAN1 reset
Set and cleared by software.
0: does not reset CAN1
1: resets CAN1

Bit 24 Reserved, must be kept at reset value.

Bit 23 **I2C3RST:** I2C3 reset
Set and cleared by software.
0: does not reset I2C3
1: resets I2C3

Bit 22 **I2C2RST:** I2C2 reset
Set and cleared by software.
0: does not reset I2C2
1: resets I2C2

Bit 21 **I2C1RST:** I2C1 reset
Set and cleared by software.
0: does not reset I2C1
1: resets I2C1

Bit 20 **UART5RST:** UART5 reset
Set and cleared by software.
0: does not reset UART5
1: resets UART5

Bit 19 **UART4RST:** USART4 reset
Set and cleared by software.
0: does not reset UART4
1: resets UART4

Bit 18 **USART3RST:** USART3 reset
Set and cleared by software.
0: does not reset USART3
1: resets USART3

Bit 17 **USART2RST:** USART2 reset
Set and cleared by software.
0: does not reset USART2
1: resets USART2

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3RST:** SPI3 reset
Set and cleared by software.
0: does not reset SPI3
1: resets SPI3

Bit 14 **SPI2RST:** SPI2 reset
Set and cleared by software.
0: does not reset SPI2
1: resets SPI2

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

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

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

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

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

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

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

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

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

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

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

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

7.3.9 RCC APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x24

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TIM11 RST	TIM10 RST	TIM9 RST
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG RST	Reser-ved	SPI1 RST	SDIO RST	Reserved		ADC RST	Reserved		USART 6 RST	USART 1 RST	Reserved		TIM8 RST	TIM1 RST
	rw		rw	rw			rw			rw	rw			rw	rw

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

Bit 18 **TIM11RST:** TIM11 reset
Set and cleared by software.
0: does not reset TIM11
1: resets TIM14

Bit 17 **TIM10RST:** TIM10 reset
Set and cleared by software.
0: does not reset TIM10
1: resets TIM10

Bit 16 **TIM9RST:** TIM9 reset
Set and cleared by software.
0: does not reset TIM9
1: resets TIM9

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SYSCFGRST:** System configuration controller reset
Set and cleared by software.
0: does not reset the System configuration controller
1: resets the System configuration controller

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1RST:** SPI1 reset
Set and cleared by software.
0: does not reset SPI1
1: resets SPI1

Bit 11 **SDIORST:** SDIO reset
Set and cleared by software.
0: does not reset the SDIO module
1: resets the SDIO module

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

Bit 8 **ADCRST:** ADC interface reset (common to all ADCs)
Set and cleared by software.
0: does not reset the ADC interface
1: resets the ADC interface

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

Bit 5 **USART6RST:** USART6 reset
Set and cleared by software.
0: does not reset USART6
1: resets USART6

Bit 4 **USART1RST:** USART1 reset
Set and cleared by software.
0: does not reset USART1
1: resets USART1

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

Bit 1 **TIM8RST:** TIM8 reset
Set and cleared by software.
0: does not reset TIM8
1: resets TIM8

Bit 0 **TIM1RST:** TIM1 reset
Set and cleared by software.
0: does not reset TIM1
1: resets TIM1

7.3.10 RCC AHB1 peripheral clock enable register (RCC_AHB1ENR)

Address offset: 0x30

Reset value: 0x0010 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reser-ved	OTGHS ULPIEN	OTGHSEN	ETHMACPTPEN	ETHMACRXEN	ETHMACTXEN	ETHMACEN	Reserved		DMA2EN	DMA1EN	CCMDATARAMEN	Res.	BKPSRAMEN	Reserved	
	rw	rw	rw	rw	rw	rw			rw	rw			rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CRCEN	Reserved			GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **OTGHSULPIEN:** USB OTG HSULPI clock enable
Set and cleared by software. This bit must be cleared when the OTG_HS is used in FS mode.
0: USB OTG HS ULPI clock disabled
1: USB OTG HS ULPI clock enabled

Bit 29 **OTGHSEN:** USB OTG HS clock enable
Set and cleared by software.
0: USB OTG HS clock disabled
1: USB OTG HS clock enabled

Bit 28 **ETHMACPTPEN:** Ethernet PTP clock enable
Set and cleared by software.
0: Ethernet PTP clock disabled
1: Ethernet PTP clock enabled

Bit 27 **ETHMACRXEN:** Ethernet Reception clock enable
Set and cleared by software.
0: Ethernet Reception clock disabled
1: Ethernet Reception clock enabled

Bit 26 **ETHMACTXEN:** Ethernet Transmission clock enable
Set and cleared by software.
0: Ethernet Transmission clock disabled
1: Ethernet Transmission clock enabled

Bit 25 **ETHMACEN:** Ethernet MAC clock enable
Set and cleared by software.
0: Ethernet MAC clock disabled
1: Ethernet MAC clock enabled

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

Bit 22 **DMA2EN:** DMA2 clock enable
Set and cleared by software.
0: DMA2 clock disabled
1: DMA2 clock enabled

Bit 21 **DMA1EN:** DMA1 clock enable
Set and cleared by software.
0: DMA1 clock disabled
1: DMA1 clock enabled

Bit 20 **CCMDATARAMEN**: CCM data RAM clock enable
Set and cleared by software.
0: CCM data RAM clock disabled
1: CCM data RAM clock enabled

Bit 19 Reserved, must be kept at reset value.

Bit 18 **BKPSRAMEN:** Backup SRAM interface clock enable
Set and cleared by software.
0: Backup SRAM interface clock disabled
1: Backup SRAM interface clock enabled

Bits 17:13 Reserved, must be kept at reset value.

Bit 12 **CRCEN:** CRC clock enable
Set and cleared by software.
0: CRC clock disabled
1: CRC clock enabled

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

Bit 8 **GPIOIEN:** IO port I clock enable
Set and cleared by software.
0: IO port I clock disabled
1: IO port I clock enabled

Bit 7 **GPIOHEN:** IO port H clock enable
Set and cleared by software.
0: IO port H clock disabled
1: IO port H clock enabled

Bit 6 **GPIOGEN:** IO port G clock enable
Set and cleared by software.
0: IO port G clock disabled
1: IO port G clock enabled

Bit 5 **GPIOFEN:** IO port F clock enable
Set and cleared by software.
0: IO port F clock disabled
1: IO port F clock enabled

Bit 4 **GPIOEEN:** IO port E clock enable
Set and cleared by software.
0: IO port E clock disabled
1: IO port E clock enabled

Bit 3 **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

7.3.11 RCC AHB2 peripheral clock enable register (RCC_AHB2ENR)

Address offset: 0x34

Reset value: 0x0000 0000

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

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

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

Bit 7 **OTGFSEN:** USB OTG FS clock enable
Set and cleared by software.
0: USB OTG FS clock disabled
1: USB OTG FS clock enabled

Bit 6 **RNGEN:** Random number generator clock enable
Set and cleared by software.
0: Random number generator clock disabled
1: Random number generator clock enabled

Bit 5 **HASHEN:** Hash modules clock enable
Set and cleared by software.
0: Hash modules clock disabled
1: Hash modules clock enabled

Bit 4 **CRYPEN:** Cryptographic modules clock enable
Set and cleared by software.
0: cryptographic module clock disabled
1: cryptographic module clock enabled

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

Bit 0 **DCMIEN:** Camera interface enable
Set and cleared by software.
0: Camera interface clock disabled
1: Camera interface clock enabled

7.3.12 RCC AHB3 peripheral clock enable register (RCC_AHB3ENR)

Address offset: 0x38

Reset value: 0x0000 0000

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

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

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

Bit 0 **FSMCEN:** Flexible static memory controller module clock enable
Set and cleared by software.
0: FSMC module clock disabled
1: FSMC module clock enabled

7.3.13 RCC APB1 peripheral clock enable register (RCC_APB1ENR)

Address offset: 0x40

Reset value: 0x0000 0000

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

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

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

Bit 29 **DACEN:** DAC interface clock enable
Set and cleared by software.
0: DAC interface clock disabled
1: DAC interface clock enable

Bit 28 **PWREN:** Power interface clock enable
Set and cleared by software.
0: Power interface clock disabled
1: Power interface clock enable

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CAN2EN:** CAN 2 clock enable
Set and cleared by software.
0: CAN 2 clock disabled
1: CAN 2 clock enabled

Bit 25 **CAN1EN:** CAN 1 clock enable
Set and cleared by software.
0: CAN 1 clock disabled
1: CAN 1 clock enabled

Bit 24 Reserved, must be kept at reset value.

Bit 23 **I2C3EN:** I2C3 clock enable
Set and cleared by software.
0: I2C3 clock disabled
1: I2C3 clock enabled

Bit 22 **I2C2EN:** I2C2 clock enable
Set and cleared by software.
0: I2C2 clock disabled
1: I2C2 clock enabled

Bit 21 **I2C1EN:** I2C1 clock enable
Set and cleared by software.
0: I2C1 clock disabled
1: I2C1 clock enabled

Bit 20 **UART5EN:** UART5 clock enable
Set and cleared by software.
0: UART5 clock disabled
1: UART5 clock enabled

Bit 19 **UART4EN:** UART4 clock enable
Set and cleared by software.
0: UART4 clock disabled
1: UART4 clock enabled

Bit 18 **USART3EN:** USART3 clock enable
Set and cleared by software.
0: USART3 clock disabled
1: USART3 clock enabled

Bit 17 **USART2EN:** USART2 clock enable
Set and cleared by software.
0: USART2 clock disabled
1: USART2 clock enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3EN:** SPI3 clock enable
Set and cleared by software.
0: SPI3 clock disabled
1: SPI3 clock enabled

Bit 14 **SPI2EN:** SPI2 clock enable
Set and cleared by software.
0: SPI2 clock disabled
1: SPI2 clock enabled

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

Bit 11 **WWDGEN:** Window watchdog clock enable
Set and cleared by software.
0: Window watchdog clock disabled
1: Window watchdog clock enabled

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

Bit 8 **TIM14EN:** TIM14 clock enable
Set and cleared by software.
0: TIM14 clock disabled
1: TIM14 clock enabled

Bit 7 **TIM13EN:** TIM13 clock enable
Set and cleared by software.
0: TIM13 clock disabled
1: TIM13 clock enabled

Bit 6 **TIM12EN:** TIM12 clock enable
Set and cleared by software.
0: TIM12 clock disabled
1: TIM12 clock enabled

Bit 5 **TIM7EN:** TIM7 clock enable
Set and cleared by software.
0: TIM7 clock disabled
1: TIM7 clock enabled

Bit 4 **TIM6EN:** TIM6 clock enable
Set and cleared by software.
0: TIM6 clock disabled
1: TIM6 clock enabled

Bit 3 **TIM5EN:** TIM5 clock enable
Set and cleared by software.
0: TIM5 clock disabled
1: TIM5 clock enabled

Bit 2 **TIM4EN:** TIM4 clock enable
Set and cleared by software.
0: TIM4 clock disabled
1: TIM4 clock enabled

Bit 1 **TIM3EN:** TIM3 clock enable
Set and cleared by software.
0: TIM3 clock disabled
1: TIM3 clock enabled

Bit 0 **TIM2EN:** TIM2 clock enable
Set and cleared by software.
0: TIM2 clock disabled
1: TIM2 clock enabled

7.3.14 RCC APB2 peripheral clock enable register (RCC_APB2ENR)

Address offset: 0x44

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TIM11 EN	TIM10 EN	TIM9 EN
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG EN	Reser-ved	SPI1 EN	SDIO EN	ADC3 EN	ADC2 EN	ADC1 EN	Reserved		USART 6 EN	USART 1 EN	Reserved		TIM8 EN	TIM1 EN
	rw		rw	rw	rw	rw	rw			rw	rw			rw	rw

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

Bit 18 **TIM11EN:** TIM11 clock enable
Set and cleared by software.
0: TIM11 clock disabled
1: TIM11 clock enabled

Bit 17 **TIM10EN:** TIM10 clock enable
Set and cleared by software.
0: TIM10 clock disabled
1: TIM10 clock enabled

Bit 16 **TIM9EN:** TIM9 clock enable
Set and cleared by software.
0: TIM9 clock disabled
1: TIM9 clock enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SYSCFGEN:** System configuration controller clock enable
Set and cleared by software.
0: System configuration controller clock disabled
1: System configuration controller clock enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1EN:** SPI1 clock enable
Set and cleared by software.
0: SPI1 clock disabled
1: SPI1 clock enabled

Bit 11 **SDIOEN:** SDIO clock enable
Set and cleared by software.
0: SDIO module clock disabled
1: SDIO module clock enabled

Bit 10 **ADC3EN:** ADC3 clock enable
Set and cleared by software.
0: ADC3 clock disabled
1: ADC3 clock disabled

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

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

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

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

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

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

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

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

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

Address offset: 0x50

Reset value: 0x7E67 91FF

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reser-ved	OTGHSULPILPEN	OTGHSLPEN	ETHPTPLPEN	ETHRXLPEN	ETHTXLPEN	ETHMACLPEN	Reserved		DMA2LPEN	DMA1LPEN	Reserved		BKPSRAMLPEN	SRAM2LPEN	SRAM1LPEN
	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLITFLPEN	Reserved		CRCLPEN	Reserved			GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
rw			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

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

Set and cleared by software. This bit must be cleared when the OTG_HS is used in FS mode.

0: USB OTG HS ULPI clock disabled during Sleep mode

1: USB OTG HS ULPI clock enabled during Sleep mode

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

Set and cleared by software.

0: USB OTG HS clock disabled during Sleep mode

1: USB OTG HS clock enabled during Sleep mode

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

Set and cleared by software.

0: Ethernet PTP clock disabled during Sleep mode

1: Ethernet PTP clock enabled during Sleep mode

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

Set and cleared by software.

0: Ethernet reception clock disabled during Sleep mode

1: Ethernet reception clock enabled during Sleep mode

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

Set and cleared by software.

0: Ethernet transmission clock disabled during sleep mode

1: Ethernet transmission clock enabled during sleep mode

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

Set and cleared by software.

0: Ethernet MAC clock disabled during Sleep mode

1: Ethernet MAC clock enabled during Sleep mode

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

Bit 22 **DMA2LPEN:** DMA2 clock enable during Sleep mode
Set and cleared by software.
0: DMA2 clock disabled during Sleep mode
1: DMA2 clock enabled during Sleep mode

Bit 21 **DMA1LPEN:** DMA1 clock enable during Sleep mode
Set and cleared by software.
0: DMA1 clock disabled during Sleep mode
1: DMA1 clock enabled during Sleep mode

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **BKPSRAMLPEN:** Backup SRAM interface clock enable during Sleep mode
Set and cleared by software.
0: Backup SRAM interface clock disabled during Sleep mode
1: Backup SRAM interface clock enabled during Sleep mode

Bit 17 **SRAM2LPEN:** SRAM 2 interface clock enable during Sleep mode
Set and cleared by software.
0: SRAM 2 interface clock disabled during Sleep mode
1: SRAM 2 interface clock enabled during Sleep mode

Bit 16 **SRAM1LPEN:** SRAM 1interface clock enable during Sleep mode
Set and cleared by software.
0: SRAM 1 interface clock disabled during Sleep mode
1: SRAM 1 interface clock enabled during Sleep mode

Bit 15 **FLITFLPEN:** Flash interface clock enable during Sleep mode
Set and cleared by software.
0: Flash interface clock disabled during Sleep mode
1: Flash interface clock enabled during Sleep mode

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

Bit 12 **CRCLPEN:** CRC clock enable during Sleep mode
Set and cleared by software.
0: CRC clock disabled during Sleep mode
1: CRC clock enabled during Sleep mode

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

Bit 8 **GPIOILPEN:** IO port I clock enable during Sleep mode
Set and cleared by software.
0: IO port I clock disabled during Sleep mode
1: IO port I clock enabled during Sleep mode

Bit 7 **GPIOHLPEN:** IO port H clock enable during Sleep mode
Set and cleared by software.
0: IO port H clock disabled during Sleep mode
1: IO port H clock enabled during Sleep mode

Bits 6 **GPIOGLPEN:** IO port G clock enable during Sleep mode
Set and cleared by software.
0: IO port G clock disabled during Sleep mode
1: IO port G clock enabled during Sleep mode

Bit 5 **GPIOFLPEN:** IO port F clock enable during Sleep mode
Set and cleared by software.
0: IO port F clock disabled during Sleep mode
1: IO port F clock enabled during Sleep mode

Bit 4 **GPIOELPEN:** IO port E clock enable during Sleep mode
Set and cleared by software.
0: IO port E clock disabled during Sleep mode
1: IO port E clock enabled during Sleep mode

Bit 3 **GPIODLPEN:** IO port D clock enable during Sleep mode
Set and cleared by software.
0: IO port D clock disabled during Sleep mode
1: IO port D clock enabled during Sleep mode

Bit 2 **GPIOCLPEN:** IO port C clock enable during Sleep mode
Set and cleared by software.
0: IO port C clock disabled during Sleep mode
1: IO port C clock enabled during Sleep mode

Bit 1 **GPIOBLPEN:** IO port B clock enable during Sleep mode
Set and cleared by software.
0: IO port B clock disabled during Sleep mode
1: IO port B clock enabled during Sleep mode

Bit 0 **GPIOALPEN:** IO port A clock enable during sleep mode
Set and cleared by software.
0: IO port A clock disabled during Sleep mode
1: IO port A clock enabled during Sleep mode

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

Address offset: 0x54

Reset value: 0x0000 00F1

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

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

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

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

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

Bit 5 **HASHLPEN:** Hash modules clock enable during Sleep mode
Set and cleared by software.
0: Hash modules clock disabled during Sleep mode
1: Hash modules clock enabled during Sleep mode

Bit 4 **CRYPLPEN:** Cryptography modules clock enable during Sleep mode
Set and cleared by software.
0: cryptography modules clock disabled during Sleep mode
1: cryptography modules clock enabled during Sleep mode

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

Bit 0 **DCMILPEN:** Camera interface enable during Sleep mode
Set and cleared by software.
0: Camera interface clock disabled during Sleep mode
1: Camera interface clock enabled during Sleep mode

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

Address offset: 0x58

Reset value: 0x0000 0001

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

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

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

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

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

Address offset: 0x60

Reset value: 0x36FE C9FF

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DAC LPEN	PWR LPEN	RESERVED	CAN2 LPEN	CAN1 LPEN	Reser-ved	I2C3 LPEN	I2C2 LPEN	I2C1 LPEN	UART5 LPEN	UART4 LPEN	USART 3 LPEN	USART 2 LPEN	Reser-ved
		rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 LPEN	SPI2 LPEN	Reserved		WWDG LPEN	Reserved		TIM14 LPEN	TIM13 LPEN	TIM12 LPEN	TIM7 LPEN	TIM6 LPEN	TIM5 LPEN	TIM4 LPEN	TIM3 LPEN	TIM2 LPEN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 29 **DACLPEN:** DAC interface clock enable during Sleep mode
Set and cleared by software.
0: DAC interface clock disabled during Sleep mode
1: DAC interface clock enabled during Sleep mode

Bit 28 **PWRLPEN:** Power interface clock enable during Sleep mode
Set and cleared by software.
0: Power interface clock disabled during Sleep mode
1: Power interface clock enabled during Sleep mode

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CAN2LPEN:** CAN 2 clock enable during Sleep mode
Set and cleared by software.
0: CAN 2 clock disabled during sleep mode
1: CAN 2 clock enabled during sleep mode

Bit 25 **CAN1LPEN:** CAN 1 clock enable during Sleep mode
Set and cleared by software.
0: CAN 1 clock disabled during Sleep mode
1: CAN 1 clock enabled during Sleep mode

Bit 24 Reserved, must be kept at reset value.

Bit 23 **I2C3LPEN:** I2C3 clock enable during Sleep mode
Set and cleared by software.
0: I2C3 clock disabled during Sleep mode
1: I2C3 clock enabled during Sleep mode

Bit 22 **I2C2LPEN:** I2C2 clock enable during Sleep mode
Set and cleared by software.
0: I2C2 clock disabled during Sleep mode
1: I2C2 clock enabled during Sleep mode

Bit 21 **I2C1LPEN:** I2C1 clock enable during Sleep mode
Set and cleared by software.
0: I2C1 clock disabled during Sleep mode
1: I2C1 clock enabled during Sleep mode

Bit 20 **UART5LPEN:** UART5 clock enable during Sleep mode
Set and cleared by software.
0: UART5 clock disabled during Sleep mode
1: UART5 clock enabled during Sleep mode

Bit 19 **UART4LPEN:** UART4 clock enable during Sleep mode
Set and cleared by software.
0: UART4 clock disabled during Sleep mode
1: UART4 clock enabled during Sleep mode

Bit 18 **USART3LPEN:** USART3 clock enable during Sleep mode
Set and cleared by software.
0: USART3 clock disabled during Sleep mode
1: USART3 clock enabled during Sleep mode

Bit 17 **USART2LPEN:** USART2 clock enable during Sleep mode
Set and cleared by software.
0: USART2 clock disabled during Sleep mode
1: USART2 clock enabled during Sleep mode

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3LPEN:** SPI3 clock enable during Sleep mode
Set and cleared by software.
0: SPI3 clock disabled during Sleep mode
1: SPI3 clock enabled during Sleep mode

Bit 14 **SPI2LPEN:** SPI2 clock enable during Sleep mode
Set and cleared by software.
0: SPI2 clock disabled during Sleep mode
1: SPI2 clock enabled during Sleep mode

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

Bit 11 **WWDGLPEN:** Window watchdog clock enable during Sleep mode
Set and cleared by software.
0: Window watchdog clock disabled during sleep mode
1: Window watchdog clock enabled during sleep mode

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

Bit 8 **TIM14LPEN:** TIM14 clock enable during Sleep mode
Set and cleared by software.
0: TIM14 clock disabled during Sleep mode
1: TIM14 clock enabled during Sleep mode

Bit 7 **TIM13LPEN:** TIM13 clock enable during Sleep mode
Set and cleared by software.
0: TIM13 clock disabled during Sleep mode
1: TIM13 clock enabled during Sleep mode

Bit 6 **TIM12LPEN:** TIM12 clock enable during Sleep mode
Set and cleared by software.
0: TIM12 clock disabled during Sleep mode
1: TIM12 clock enabled during Sleep mode

Bit 5 **TIM7LPEN:** TIM7 clock enable during Sleep mode
Set and cleared by software.
0: TIM7 clock disabled during Sleep mode
1: TIM7 clock enabled during Sleep mode

Bit 4 **TIM6LPEN:** TIM6 clock enable during Sleep mode
Set and cleared by software.
0: TIM6 clock disabled during Sleep mode
1: TIM6 clock enabled during Sleep mode

Bit 3 **TIM5LPEN:** TIM5 clock enable during Sleep mode
Set and cleared by software.
0: TIM5 clock disabled during Sleep mode
1: TIM5 clock enabled during Sleep mode

Bit 2 **TIM4LPEN:** TIM4 clock enable during Sleep mode
Set and cleared by software.
0: TIM4 clock disabled during Sleep mode
1: TIM4 clock enabled during Sleep mode

Bit 1 **TIM3LPEN:** TIM3 clock enable during Sleep mode
Set and cleared by software.
0: TIM3 clock disabled during Sleep mode
1: TIM3 clock enabled during Sleep mode

Bit 0 **TIM2LPEN:** TIM2 clock enable during Sleep mode
Set and cleared by software.
0: TIM2 clock disabled during Sleep mode
1: TIM2 clock enabled during Sleep mode

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

Address offset: 0x64

Reset value: 0x0007 5F33

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

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

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

Bit 18 **TIM11LPEN:** TIM11 clock enable during Sleep mode
Set and cleared by software.
0: TIM11 clock disabled during Sleep mode
1: TIM11 clock enabled during Sleep mode

Bit 17 **TIM10LPEN:** TIM10 clock enable during Sleep mode
Set and cleared by software.
0: TIM10 clock disabled during Sleep mode
1: TIM10 clock enabled during Sleep mode

Bit 16 **TIM9LPEN:** TIM9 clock enable during sleep mode
Set and cleared by software.
0: TIM9 clock disabled during Sleep mode
1: TIM9 clock enabled during Sleep mode

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SYSCFGLPEN:** System configuration controller clock enable during Sleep mode
Set and cleared by software.
0: System configuration controller clock disabled during Sleep mode
1: System configuration controller clock enabled during Sleep mode

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1LPEN:** SPI1 clock enable during Sleep mode
Set and cleared by software.
0: SPI1 clock disabled during Sleep mode
1: SPI1 clock enabled during Sleep mode

Bit 11 **SDIOLPEN:** SDIO clock enable during Sleep mode
Set and cleared by software.
0: SDIO module clock disabled during Sleep mode
1: SDIO module clock enabled during Sleep mode

Bit 10 **ADC3LPEN:** ADC 3 clock enable during Sleep mode
Set and cleared by software.
0: ADC 3 clock disabled during Sleep mode
1: ADC 3 clock disabled during Sleep mode

Bit 9 **ADC2LPEN:** ADC2 clock enable during Sleep mode
Set and cleared by software.
0: ADC2 clock disabled during Sleep mode
1: ADC2 clock disabled during Sleep mode

Bit 8 **ADC1LPEN:** ADC1 clock enable during Sleep mode
Set and cleared by software.
0: ADC1 clock disabled during Sleep mode
1: ADC1 clock disabled during Sleep mode

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

Bit 5 **USART6LPEN:** USART6 clock enable during Sleep mode
Set and cleared by software.
0: USART6 clock disabled during Sleep mode
1: USART6 clock enabled during Sleep mode

Bit 4 **USART1LPEN:** USART1 clock enable during Sleep mode
Set and cleared by software.
0: USART1 clock disabled during Sleep mode
1: USART1 clock enabled during Sleep mode

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

Bit 1 **TIM8LPEN:** TIM8 clock enable during Sleep mode
Set and cleared by software.
0: TIM8 clock disabled during Sleep mode
1: TIM8 clock enabled during Sleep mode

Bit 0 **TIM1LPEN:** TIM1 clock enable during Sleep mode
Set and cleared by software.
0: TIM1 clock disabled during Sleep mode
1: TIM1 clock enabled during Sleep mode

7.3.20 RCC Backup domain control register (RCC_BDCR)

Address offset: 0x70

Reset value: 0x0000 0000, reset by Backup domain reset.
Access: 0 ≤ wait state ≤ 3, word, half-word and byte access
Wait states are inserted in case of successive accesses to this register.

The LSEON, LSEBYP, RTCSEL and RTCEN bits in the *RCC Backup domain control register (RCC_BDCR)* are in the Backup domain. As a result, after Reset, these bits are write-protected and the DBP bit in the *PWR power control register (PWR_CR) for STM32F405xx/07xx and STM32F415xx/17xx* has to be set before these can be modified. Refer to *Section 7.1.1: System reset on page 213* for further information. These bits are only reset after a Backup domain Reset (see *Section 7.1.3: Backup domain reset*). Any internal or external Reset will not have any effect on these bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															BDRST
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Reserved					RTCSEL[1:0]		Reserved					LSEBYP	LSERDY	LSEON
rw						rw	rw						rw	r	rw

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

Bit 16 **BDRST:** Backup domain software reset
Set and cleared by software.
0: Reset not activated
1: Resets the entire Backup domain
Note: The BKPSRAM is not affected by this reset, the only way of resetting the BKPSRAM is through the Flash interface when a protection level change from level 1 to level 0 is requested.

Bit 15 **RTCEN:** RTC clock enable
Set and cleared by software.
0: RTC clock disabled
1: RTC clock enabled

Bits 14:10 Reserved, must be kept at reset value.

Bits 9:8 **RTCSEL[1:0]:** RTC clock source selection
Set by software to select the clock source for the RTC. Once the RTC clock source has been selected, it cannot be changed anymore unless the Backup domain is reset. The BDRST bit can be used to reset them.
00: No clock
01: LSE oscillator clock used as the RTC clock
10: LSI oscillator clock used as the RTC clock
11: HSE oscillator clock divided by a programmable prescaler (selection through the RTCPRE[4:0] bits in the RCC clock configuration register (RCC_CFGR)) used as the RTC clock

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

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

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

0: LSE oscillator not bypassed

1: LSE oscillator bypassed

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

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

0: LSE clock not ready

1: LSE clock ready

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

Set and cleared by software.

0: LSE clock OFF

1: LSE clock ON

7.3.21 RCC clock control & status register (RCC_CSR)

Address offset: 0x74

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

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	BORRS TF	RMVF	Reserved							
r	r	r	r	r	r	r	rt_w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														LSIRDY	LSION
														r	rw

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

Set by hardware when a Low-power management reset occurs.
Cleared by writing to the RMVF bit.

0: No Low-power management reset occurred

1: Low-power management reset occurred

For further information on Low-power management reset, refer to *Low-power management reset*.

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

Set by hardware when a window watchdog reset occurs.
Cleared by writing to the RMVF bit.

0: No window watchdog reset occurred

1: Window watchdog reset occurred

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

Set by hardware when an independent watchdog reset from V_{DD} domain occurs.
Cleared by writing to the RMVF bit.

0: No watchdog reset occurred

1: Watchdog reset occurred

Bit 28 **SFTRSTF:** Software reset flag
Set by hardware when a software reset occurs.
Cleared by writing to the RMVF bit.
0: No software reset occurred
1: Software reset occurred

Bit 27 **PORRSTF:** POR/PDR reset flag
Set by hardware when a POR/PDR reset occurs.
Cleared by writing to the RMVF bit.
0: No POR/PDR reset occurred
1: POR/PDR reset occurred

Bit 26 **PINRSTF:** PIN reset flag
Set by hardware when a reset from the NRST pin occurs.
Cleared by writing to the RMVF bit.
0: No reset from NRST pin occurred
1: Reset from NRST pin occurred

Bit 25 **BORRSTF:** BOR reset flag
Cleared by software by writing the RMVF bit.
Set by hardware when a POR/PDR or BOR reset occurs.
0: No POR/PDR or BOR reset occurred
1: POR/PDR or BOR reset occurred

Bit 24 **RMVF:** Remove reset flag
Set by software to clear the reset flags.
0: No effect
1: Clear the reset flags

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

Bit 1 **LSIRDY:** Internal low-speed oscillator ready
Set and cleared by hardware to indicate when the internal RC 40 kHz oscillator is stable. After the LSION bit is cleared, LSIRDY goes low after 3 LSI clock cycles.
0: LSI RC oscillator not ready
1: LSI RC oscillator ready

Bit 0 **LSION:** Internal low-speed oscillator enable
Set and cleared by software.
0: LSI RC oscillator OFF
1: LSI RC oscillator ON

7.3.22 RCC spread spectrum clock generation register (RCC_SSCGR)

Address offset: 0x80

Reset value: 0x0000 0000

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

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

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

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

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

Bit 31 **SSCGEN:** Spread spectrum modulation enable
Set and cleared by software.
0: Spread spectrum modulation DISABLE. (To write after clearing CR[24]=PLLON bit)
1: Spread spectrum modulation ENABLE. (To write before setting CR[24]=PLLON bit)

Bit 30 **SPREADSEL:** Spread Select
Set and cleared by software.
To write before to set CR[24]=PLLON bit.
0: Center spread
1: Down spread

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

Bits 27:13 **INCSTEP:** Incrementation step
Set and cleared by software. To write before setting CR[24]=PLLON bit.
Configuration input for modulation profile amplitude.

Bits 12:0 **MODPER:** Modulation period
Set and cleared by software. To write before setting CR[24]=PLLON bit.
Configuration input for modulation profile period.

7.3.23 RCC PLLI2S configuration register (RCC_PLLI2SCFGR)

Address offset: 0x84

Reset value: 0x2000 3000

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

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

- $f_{(VCO\ clock)} = f_{(PLLI2S\ clock\ input)} \times (PLLI2SN / PLLM)$
- $f_{(PLL\ I2S\ clock\ output)} = f_{(VCO\ clock)} / PLLI2SR$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	PLLI2SR2	PLLI2SR1	PLLI2SR0	Reserved											
	rw	rw	rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLLI2SN8	PLLI2SN7	PLLI2SN6	PLLI2SN5	PLLI2SN4	PLLI2SN3	PLLI2SN2	PLLI2SN1	PLLI2SN0	Reserved					
	rw	rw	rw	rw	rw	rw	rw	rw	rw						

Bit 31 Reserved, must be kept at reset value.

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

Set and cleared by software to control the I2S clock frequency. These bits should be written only if the PLLI2S is disabled. The factor must be chosen in accordance with the prescaler values inside the I2S peripherals, to reach 0.3% error when using standard crystals and 0% error with audio crystals. For more information about I2S clock frequency and precision, refer to *Section 28.4.4: Clock generator* in the I2S chapter.

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

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

000: PLLR = 0, wrong configuration

001: PLLR = 1, wrong configuration

010: PLLR = 2

...

111: PLLR = 7

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

Bits 14:6 **PLLI2SN:** PLLI2S multiplication factor for VCO

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

Caution: The software has to set these bits correctly to ensure that the VCO output frequency is between 100 and 432 MHz.

VCO output frequency = VCO input frequency × PLLI2SN with 50 ≤PLLI2SN ≤432

000000000: PLLI2SN = 0, wrong configuration

000000001: PLLI2SN = 1, wrong configuration

...

000110010: PLLI2SN = 50

...

001100011: PLLI2SN = 99

001100100: PLLI2SN = 100

001100101: PLLI2SN = 101

001100110: PLLI2SN = 102

...

110110000: PLLI2SN = 432

110110001: PLLI2SN = 433, wrong configuration

...

111111111: PLLI2SN = 511, wrong configuration

Note: Multiplication factors ranging from 50 and 99 are possible for VCO input frequency higher than 1 MHz. However care must be taken that the minimum VCO output frequency respects the value specified above.

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

7.3.24 RCC register map

Table 34 gives the register map and reset values.

Table 34. RCC register map and reset values

Addr. 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	RCC_CR	Reserved				PLL I2SRDY	PLL I2SON	PLL RDY	PLL ON	Reserved				CSSON	HSEBYP	HSERDY	HSEON	HSICAL 7	HSICAL 6	HSICAL 5	HSICAL 4	HSICAL 3	HSICAL 2	HSICAL 1	HSICAL 0	HSITRIM 4	HSITRIM 3	HSITRIM 2	HSITRIM 1	HSITRIM 0	Reserved	HSIRDY	HSION
0x04	RCC_ PLLCFGR	Reserved				PLLQ 3	PLLQ 2	PLLQ 1	PLLQ 0	Reserved	PLLSRC	Reserved				PLLP 1	PLLP 0	Reserved	PLLN 8	PLLN 7	PLLN 6	PLLN 5	PLLN 4	PLLN 3	PLLN 2	PLLN 1	PLLN 0	PLLM 5	PLLM 4	PLLM 3	PLLM 2	PLLM 1	PLLM 0
0x08	RCC_CFGR	MCO2 1	MCO2 0	MCO2PRE2	MCO2PRE1	MCO2PRE0	MCO1PRE2	MCO1PRE1	MCO1PRE0	I2SSRC	MCO1 1	MCO1 0	RTCPRE 4	RTCPRE 3	RTCPRE 2	RTCPRE 1	RTCPRE 0	PPRE2 2	PPRE2 1	PPRE2 0	PPRE1 2	PPRE1 1	PPRE1 0	Reserved		HPRE 3	HPRE 2	HPRE 1	HPRE 0	SWS 1	SWS 0	SW 1	SW 0
0x0C	RCC_CIR	Reserved								CSSC	Reserved	PLLI2SRDYC	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	Reserved	PLLI2SRDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Reserved	PLLI2SRDYF	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
0x10	RCC_ AHB1RSTR	Reserved		OTGHSRST	Reserved			ETHMACRST	Reserved	DMA2RST	DMA1RST	Reserved									CRCRST	Reserved			GPIOIRST	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
0x14	RCC_ AHB2RSTR	Reserved																							OTGFSRST	RNGRST	HSAHRST	CRYPRST	Reserved				DCMIRST
0x18	RCC_ AHB3RSTR	Reserved																															FSMCRST
0x1C	Reserved	Reserved																															
0x20	RCC_ APB1RSTR	Reserved		DACRST	PWRRST	Reserved	CAN2RST	CAN1RST	Reserved	I2C3RST	I2C2RST	I2C1RST	UART5RST	UART4RST	UART3RST	UART2RST	Reserved	SPI3RST	SPI2RST	Reserved			WWDGRST	Reserved	TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
0x24	RCC_ APB2RSTR	Reserved													TIM11RST	TIM10RST	TIM9RST	Reserved	SYSCFGRST	Reserved	SPI1RST	SDIORST	Reserved	ADCRST	Reserved		USART6RST	USART1RST	Reserved		TIM8RST	TIM1RST
0x28	Reserved	Reserved																															
0x2C	Reserved	Reserved																															
0x30	RCC_ AHB1ENR	Reserved	OTGHSULPIEN	OTGHSEN	ETHMACPTPEN	ETHMACRXEN	ETHMACTXEN	ETHMACEN	Reserved	DMA2EN	DMA1EN	CCMDATARAMEN	Reserved	BKPSRAMEN	Reserved						CRCEN	Reserved			GPIOEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
0x34	RCC_ AHB2ENR	Reserved																							OTGFSEN	RNGEN	HASHEN	CRYPEN	Reserved				DCMIEN
0x38	RCC_ AHB3ENR	Reserved																															FSMCEN
0x3C	Reserved	Reserved																															

Table 34. RCC register map and reset values (continued)

Addr. 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
0x40	RCC_ APB1ENR	Reserved		DACEN	PWREN	Reserved	CAN2EN	CAN1EN	Reserved	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Reserved	SPI3EN	SPI2EN	Reserved		WWDGEN	Reserved		TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
0x44	RCC_ APB2ENR	Reserved													TIM11EN	TIM10EN	TIM9EN	Reserved	SYSCFGEN	Reserved	SPI1EN	SDIOEN	ADC3EN	ADC2EN	ADC1EN	Reserved		USART6EN	USART1EN	Reserved		TIM8EN	TIM1EN
0x48	Reserved	Reserved																															
0x4C	Reserved	Reserved																															
0x50	RCC_AHB1LP ENR	Reserved	OTGHSULPILPEN	OTGHSLPEN	ETHMACPTPLPEN	ETHMACRXLPEN	ETHMACTXLPEN	ETHMACLPEN	Reserved	DMA2LPEN	DMA1LPEN	Reserved			BKPSRAMLPEN	SRAM2LPEN	SRAM1LPEN	FLITFLPEN	Reserved	CRCLPEN	Reserved				GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
0x54	RCC_AHB2LP ENR	Reserved																								OTGFSLPEN	RNGLPEN	HASHLPEN	CRYPLPEN	Reserved			DCMILPEN
0x58	RCC_AHB3LP ENR	Reserved																															FSMCLPEN
0x5C	Reserved	Reserved																															
0x60	RCC_APB1LP ENR	Reserved		DACLPEN	PWRLPEN	Reserved	CAN2LPEN	CAN1LPEN	Reserved	I2C3LPEN	I2C2LPEN	I2C1LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	Reserved	SPI3LPEN	SPI2LPEN	Reserved		WWDGLPEN	Reserved		TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
0x64	RCC_APB2LP ENR	Reserved													TIM11LPEN	TIM10LPEN	TIM9LPEN	Reserved	SYSCFGLPEN	Reserved	SPI1LPEN	SDIOLPEN	ADC3LPEN	ADC2LPEN	ADC1LPEN	Reserved		USART6LPEN	USART1LPEN	Reserved		TIM8LPEN	TIM1LPEN
0x68	Reserved	Reserved																															
0x6C	Reserved	Reserved																															
0x70	RCC_BDCR	Reserved															BDRST	RTCEN	Reserved				RTCSEL 1	RTCSEL 0	Reserved						LSEBYP	LSERDY	LSEON
0x74	RCC_CSR	LPWRRSTF	WWDGRSTF	WDGRSTF	SFTRSTF	PORRSTF	PADRSTF	BORRSTF	RMVF	Reserved																						LSIRDY	LSION
0x78	Reserved	Reserved																															
0x7C	Reserved	Reserved																															
0x80	RCC_SSCGR	SSCGEN	SPREADSEL	Reserved					INCSTEP														MODPER										
0x84	RCC_PLLI2SC FGR	Reserved	PLLI2SRx						Reserved										PLLI2SNx											Reserved				

Refer to *Section 2.3: Memory map* for the register boundary addresses.

8 General-purpose I/Os (GPIO)

This section applies to the whole STM32F4xx family, unless otherwise specified.

8.1 GPIO introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 32-bit set/reset register (GPIOx_BSRR), a 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection register (GPIOx_AFRH and GPIOx_AFRL).

8.2 GPIO main features

- Up to 16 I/Os under control
- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_BSRR) for bitwise write access to GPIOx_ODR
- Locking mechanism (GPIOx_LCKR) provided to freeze the I/O configuration
- Analog function
- Alternate function input/output selection registers (at most 16 AFs per I/O)
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

8.3 GPIO functional description

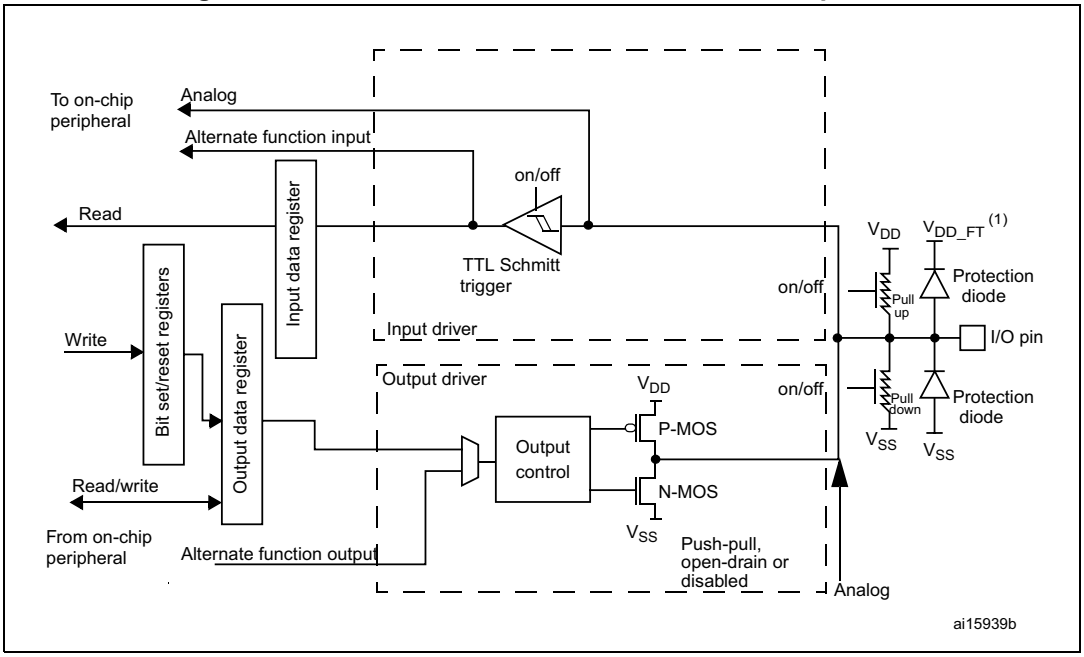
Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- 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 register is to allow atomic read/modify accesses to any of the GPIO registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

Figure 25 shows the basic structure of a 5 V tolerant I/O port bit. *Table 39* gives the possible port bit configurations.

Figure 25. Basic structure of a five-volt tolerant I/O port bit



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

Table 35. Port bit configuration table(1)

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]	PUPDR(i) [1:0]		I/O configuration	
01	0	SPEED [B:A]	0	0	GP output	PP
	0		0	1	GP output	PP + PU
	0		1	0	GP output	PP + PD
	0		1	1	Reserved	
	1		0	0	GP output	OD
	1		0	1	GP output	OD + PU
	1		1	0	GP output	OD + PD
	1		1	1	Reserved (GP output OD)	

Table 35. Port bit configuration table[1] (continued)

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]		PUPDR(i) [1:0]		I/O configuration	
10	0	SPEED [B:A]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	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 the I/O ports are configured in input floating 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

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

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

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not depending on the value in the GPIOx_PUPDR register.

8.3.2 I/O pin multiplexer and mapping

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

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

- After reset all I/Os are connected to the system's alternate function 0 (AF0)
- The peripherals' alternate functions are mapped from AF1 to AF13
- Cortex®-M4 with FPU EVENTOUT is mapped on AF15

This structure is shown in *Figure 26* below.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, proceed as follows:

- **System function**

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

Note: The user can disable some or all of the JTAG/SWD pins and so release the associated pins for GPIO usage (released pins highlighted in gray in the table).

For more details please refer to *Section 7.2.10: Clock-out capability* and *Section 6.2.10: Clock-out capability*.

Table 36. Flexible SWJ-DP pin assignment

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

- **GPIO**

  Configure the desired I/O as output or input in the GPIOx_MODER register.
- **Peripheral alternate function**

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

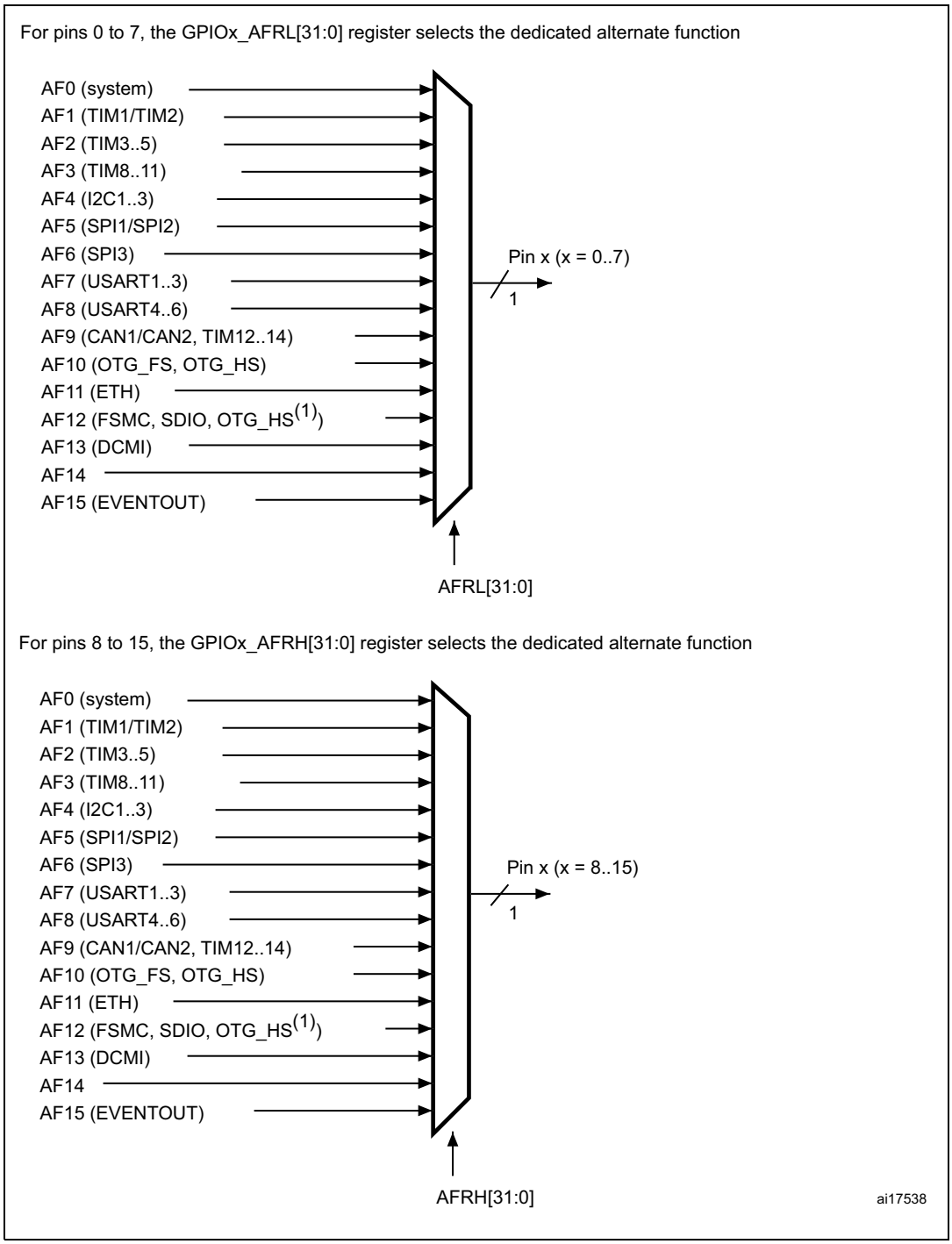
  For other peripherals:
  - Configure the desired I/O as an alternate function in the GPIOx_MODER register
  - Select the type, pull-up/pull-down and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDR registers, respectively
  - Connect the I/O to the desired AFx in the GPIOx_AFRL or GPIOx_AFRH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

  Configure the I/O pin used to output the Cortex®-M4 with FPU EVENTOUT signal by connecting it to AF15

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

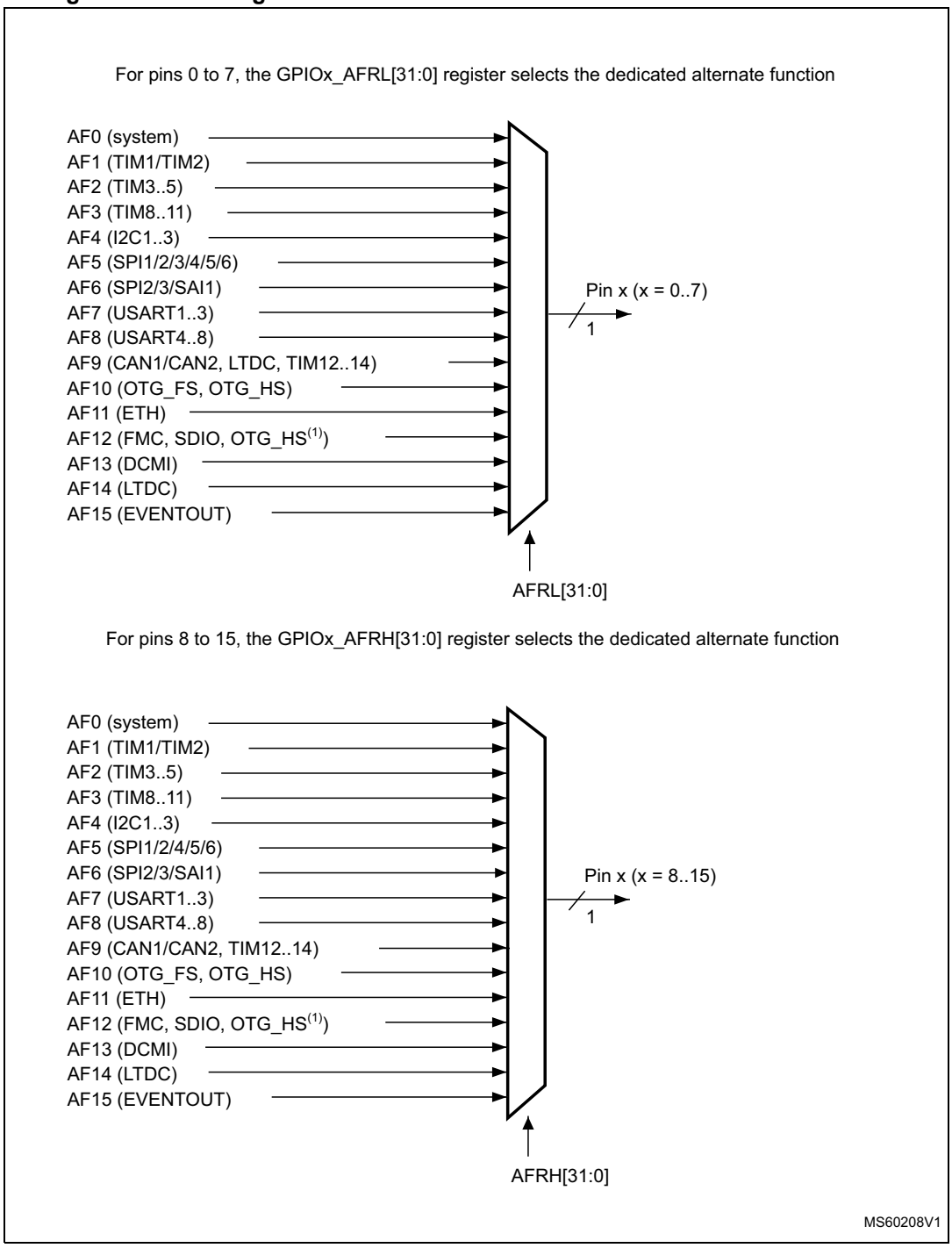
Please refer to the “Alternate function mapping” table in the datasheets for the detailed mapping of the system and peripherals’ alternate function I/O pins.

Figure 26. Selecting an alternate function on STM32F405xx/07xx and STM32F415xx/17xx



1. Configured in FS.

Figure 27. Selecting an alternate function on STM32F42xxx and STM32F43xxx



1. Configured in FS.

8.3.3 I/O port control registers

Each of the GPIOs 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 direction (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 I/O speed pins are directly connected to the corresponding GPIOx_OSPEEDR register bits whatever the I/O direction). 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..I/J/K)* and *Section 8.4.6: GPIO port output data register (GPIOx_ODR) (x = A..I/J/K)* 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: BSRR(i) and BSRR(i+SIZE). When written to 1, bit BSRR(i) sets the corresponding ODR(i) bit. When written to 1, bit BSRR(i+SIZE) resets the ODR(i) corresponding bit.

Writing any bit to 0 in 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 AHB1 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 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..I/J/K)*) can only be performed using a word (32-bit long) access to the GPIOx_LCKR register due to the fact that GPIOx_LCKR bit 16 has to be set at the same time as the [15:0] bits.

For more details please refer to LCKR register description in *Section 8.4.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A..I/J/K)*.

8.3.7 I/O alternate function input/output

Two registers are provided to select one out of the sixteen alternate function inputs/outputs available for each I/O. With these registers, you can connect an alternate function to some other pin as required by your application.
This means that a number of possible peripheral functions are multiplexed on each GPIO using the 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 one I/O.

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

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

8.3.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode, refer to *Section 12.2: External interrupt/event controller (EXTI)* and *Section 12.2.3: Wakeup event management*.

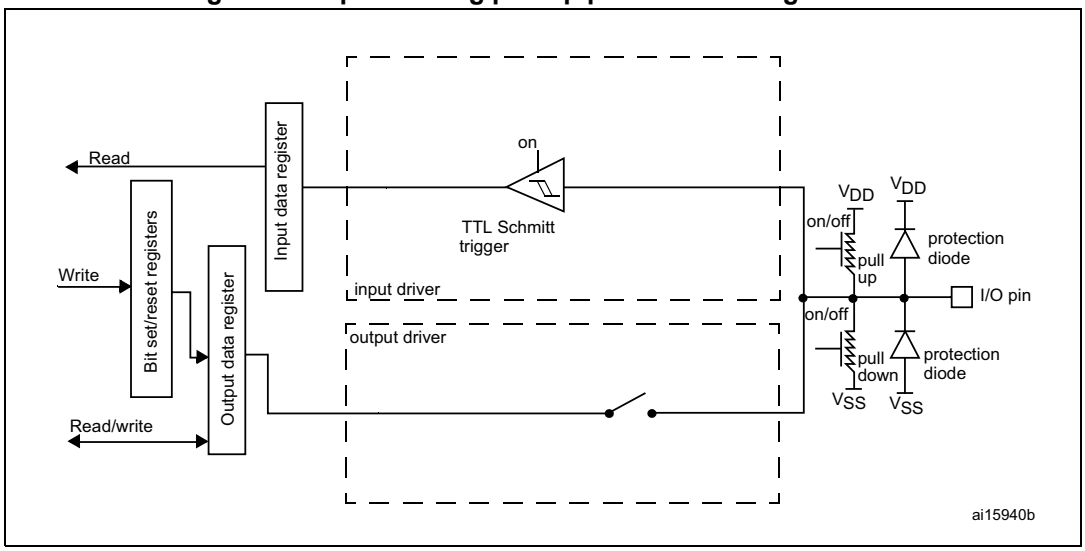
8.3.9 Input configuration

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 AHB1 clock cycle
- A read access to the input data register provides the I/O State

Figure 28 shows the input configuration of the I/O port bit.

Figure 28. Input floating/pull up/pull down configurations



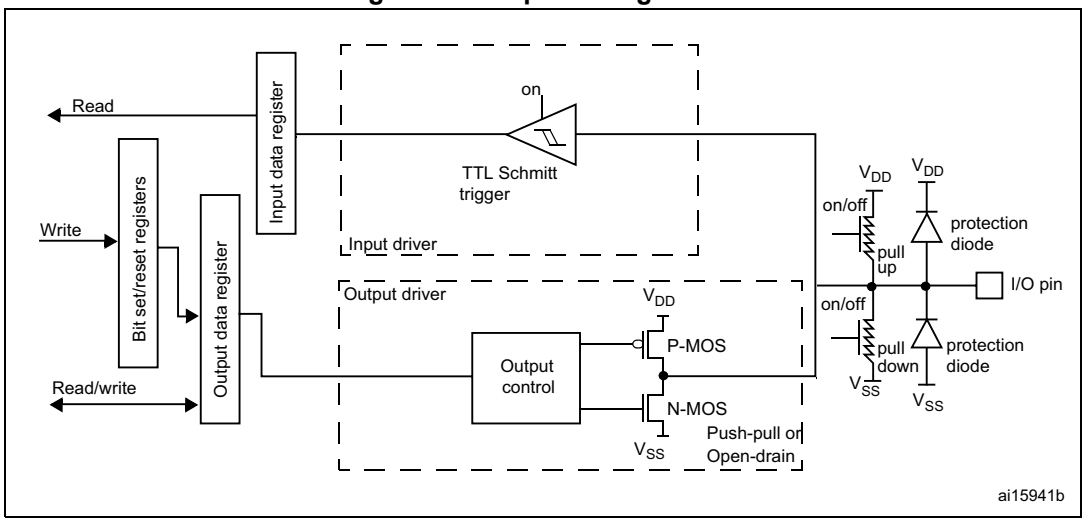
8.3.10 Output configuration

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 weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB1 clock cycle
- A read access to the input data register gets the I/O state
- A read access to the output data register gets the last written value

Figure 29 shows the output configuration of the I/O port bit.

Figure 29. Output configuration



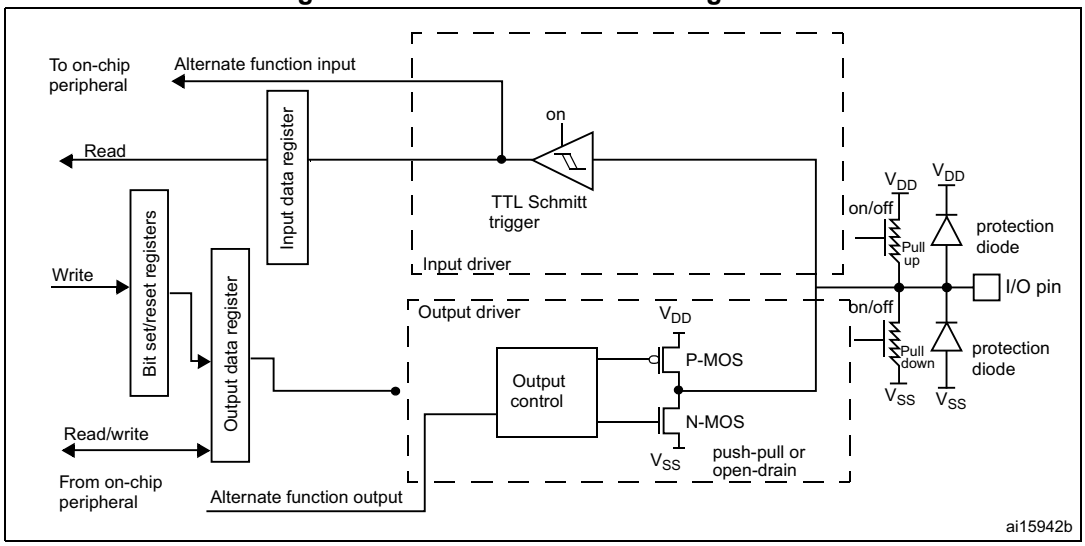
8.3.11 Alternate function configuration

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

- The output buffer can be configured as open-drain or push-pull
- The output buffer is driven by the signal 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 AHB1 clock cycle
- A read access to the input data register gets the I/O state

Figure 30 shows the Alternate function configuration of the I/O port bit.

Figure 30. Alternate function configuration



8.3.12 Analog configuration

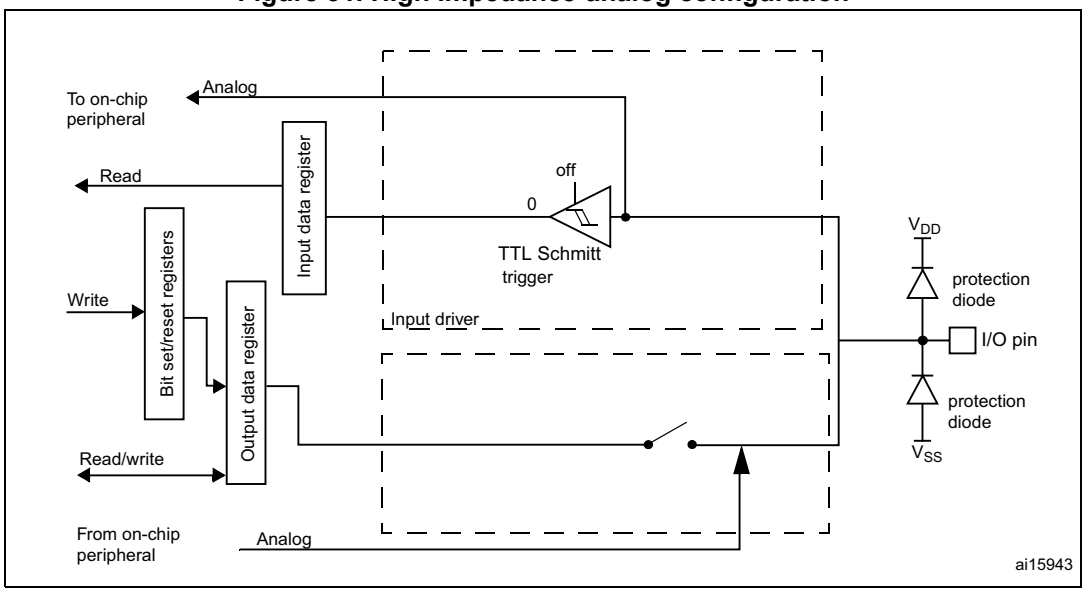
When the I/O port is programmed as analog configuration:

- 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
- Read access to the input data register gets the value “0”

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

Figure 31 shows the high-impedance, analog-input configuration of the I/O port bit.

Figure 31. High impedance-analog configuration



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

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

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

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

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

8.3.15 Selection of RTC_AF1 and RTC_AF2 alternate functions

The STM32F4xx feature two GPIO pins RTC_AF1 and RTC_AF2 that can be used for the detection of a tamper or time stamp event, or RTC_ALARM, or RTC_CALIB RTC outputs.

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

RTC_ALARM output: this output can be RTC Alarm A, RTC Alarm B or RTC Wakeup depending on the OSEL[1:0] bits in the RTC_CR register

- RTC_CALIB output: this feature is enabled by setting the COE[23] in the RTC_CR register
- RTC_TAMP1: tamper event detection
- RTC_TS: time stamp event detection

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

- RTC_TAMP1: tamper event detection
- RTC_TAMP2: tamper event detection
- RTC_TS: time stamp event detection

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

- TAMP1INSEL is used to select which pin is used as the RTC_TAMP1 tamper input
- TSINSEL is used to select which pin is used as the RTC_TS time stamp input
- ALARMOUTTYPE is used to select whether the RTC_ALARM is output in push-pull or open-drain mode

The output mechanism follows the priority order listed in *Table 37* and *Table 38*.

Table 37. RTC_AF1 pin(1)

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

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

Table 38. RTC_AF2 pin

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

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 39*.

The GPIO registers can be accessed by byte (8 bits), half-words (16 bits) or words (32 bits).

8.4.1 GPIO port mode register (GPIOx_MODER) (x = A..I/J/K)

Address offset: 0x00

Reset values:

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O direction mode.
00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

8.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A..I/J/K)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **OTy**: Port x configuration bits (y = 0..15)
These bits are written by software to configure the output type of the I/O port.
0: Output push-pull (reset state)
1: Output open-drain

8.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A..I/J/K)

Address offset: 0x08

Reset values:

- 0x0C00 0000 for port A
- 0x0000 00C0 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1 [1:0]		OSPEEDR0 1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

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 product datasheets for the values of OSPEEDRy bits versus V_{DD} range and external load.

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

Address offset: 0x0C

Reset values:

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

These bits are written by software to configure the I/O pull-up or pull-down

00: No pull-up, pull-down

01: Pull-up

10: Pull-down

11: Reserved

8.4.5 GPIO port input data register (GPIOx_IDR) (x = A..I/J/K)

Address offset: 0x10

Reset value: 0x0000 XXXX (where X means undefined)

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

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

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

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

8.4.6 GPIO port output data register (GPIOx_ODR) (x = A..I/J/K)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **ODRy**: Port output data (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..I/J/K).

8.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..I/J/K)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x reset bit y (y = 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Resets the corresponding ODRx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy:** Port x set bit y (y= 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Sets the corresponding ODRx bit

8.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A..I/J/K)

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 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 write sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

Access: 32-bit word only, read/write register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															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[16]:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.
0: Port configuration lock key not active
1: Port configuration lock key active. The GPIOx_LCKR register is locked until an MCU reset or a peripheral reset occurs.

LOCK key write sequence:
WR LCKR[16] = '1' + LCKR[15:0]
WR LCKR[16] = '0' + LCKR[15:0]
WR LCKR[16] = '1' + LCKR[15:0]
RD LCKR
RD LCKR[16] = '1' (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.
Any error in the lock sequence aborts the lock.
After the first lock sequence on any bit of the port, any read access on the LCKK bit will return '1' until the next CPU reset.

Bits 15:0 **LCKy:** Port x lock bit y (y= 0..15)

These bits are read/write but can only be written when the LCKK bit is '0.
0: Port configuration not locked
1: Port configuration locked

8.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A..I/J/K)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]			
rw	rw	rw	rw	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
AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFRLy:** Alternate function selection for port x bit y (y = 0..7)

These bits are written by software to configure alternate function I/Os

AFRLy selection:

0000: AF0
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..I/J)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]			
rw	rw	rw	rw	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
AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFRHy:** Alternate function selection for port x bit y (y = 8..15)

These bits are written by software to configure alternate function I/Os

AFRHy selection:

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 register map

The following table gives the GPIO register map and the reset values.

Table 39. GPIO register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**GPIOA_ MODER**	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	**GPIOB_ MODER**	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x00	**GPIOx_MODER** (where x = C..I/J/K)	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**GPIOx_ OTYPER** (where x = A..I/J/K)	Reserved																OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOx_ OSPEEDR** (where x = A..I/J/K except B)	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOB_ OSPEEDR**	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
0x0C	**GPIOA_PUPDR**	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**GPIOB_PUPDR**	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 39. GPIO register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	**GPIOx_PUPDR** (where x = C..I/J/K)	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**GPIOx_IDR** (where x = A..I/J/K)	Reserved																IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x14	**GPIOx_ODR** (where x = A..I/J/K)	Reserved																ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**GPIOx_BSRR** (where x = A..I/J/K)	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/J/K)	Reserved																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
0x20	**GPIOx_AFRL** (where x = A..I/J/K)	AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]				AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[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/J)	AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]				AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[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

Refer to *Section 2.3: Memory map* for the register boundary addresses.

9 System configuration controller (SYSCFG)

The system configuration controller is mainly used to remap the memory accessible in the code area, select the Ethernet PHY interface and manage the external interrupt line connection to the GPIOs.

This section applies to the whole STM32F4xx family, unless otherwise specified.

9.1 I/O compensation cell

By default the I/O compensation cell is not used. However when the I/O output buffer speed is configured in 50 MHz or 100 MHz mode, it is recommended to use the compensation cell for slew rate control on I/O $t_{f(IO)out}$/$t_{r(IO)out}$ commutation to reduce the I/O noise on power supply.

When the compensation cell is enabled, a READY flag is set to indicate that the compensation cell is ready and can be used. The I/O compensation cell can be used only when the supply voltage ranges from 2.4 to 3.6 V.

9.2 SYSCFG registers for STM32F405xx/07xx and STM32F415xx/17xx

9.2.1 SYSCFG memory remap register (SYSCFG_MEMRMP)

This register is used for specific configurations on memory remap:

- Two bits are used to configure the type of memory accessible at address 0x0000 0000. These bits are used to select the physical remap by software and so, bypass the BOOT pins.
- After reset these bits take the value selected by the BOOT pins. When booting from main Flash memory with BOOT pins set to 10 [(BOOT1,BOOT0) = (1,0)] this register takes the value 0x00.

When the FSMC is remapped at address 0x0000 0000, only the first two regions of Bank 1 memory controller (Bank1 NOR/PSRAM 1 and NOR/PSRAM 2) can be remapped. In remap mode, the CPU can access the external memory via ICode bus instead of System bus which boosts up the performance.

Address offset: 0x00

Reset value: 0x0000 000X (X is the memory mode selected by the BOOT pins)

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

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **MEM_MODE:** Memory mapping selection

Set and cleared by software. This bit controls the memory internal mapping at address 0x0000 0000. After reset these bits take the value selected by the Boot pins (except for FSMC).

00: Main Flash memory mapped at 0x0000 0000
01: System Flash memory mapped at 0x0000 0000
10: FSMC Bank1 (NOR/PSRAM 1 and 2) mapped at 0x0000 0000
11: Embedded SRAM (SRAM1) mapped at 0x0000 0000

Note: Refer to Section 2.3: Memory map for details about the memory mapping at address 0x0000 0000.

9.2.2 SYSCFG peripheral mode configuration register (SYSCFG_PMC)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

Bit 23 **MII_RMII_SEL:** Ethernet PHY interface selection

Set and Cleared by software.These bits control the PHY interface for the Ethernet MAC.

0: MII interface is selected
1: RMII PHY interface is selected

Note: This configuration must be done while the MAC is under reset and before enabling the MAC clocks.

Bits 22:0 Reserved, must be kept at reset value.

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 0 to 3)

These bits are written by software to select the source input for the EXTIx external interrupt.

0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin
0111: PH[x] pin
1000: PI[x] 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 4 to 7)

These bits are written by software to select the source input for the EXTIx external interrupt.
0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin
0111: PH[x] pin
1000: PI[x] pin

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 8 to 11)

These bits are written by software to select the source input for the EXTIx external interrupt.
0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin
0111: PH[x] pin
1000: PI[x] pin

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 12 to 15)

These bits are written by software to select the source input for the EXTIx external interrupt.

0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin
0111: PH[x] pin

Note: PI[15:12] are not used.

9.2.7 Compensation cell control register (SYSCFG_CMPCR)

Address offset: 0x20

Reset value: 0x0000 0000

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

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

Bit 8 **READY:** Compensation cell ready flag

0: I/O compensation cell not ready
1: O compensation cell ready

Bits 7:2 Reserved, must be kept at reset value.

Bit 0 **CMP_PD:** Compensation cell power-down

0: I/O compensation cell power-down mode
1: I/O compensation cell enabled

9.2.8 SYSCFG register maps for STM32F405xx/07xx and STM32F415xx/17xx

The following table gives the SYSCFG register map and the reset values.

Table 40. SYSCFG register map and reset values (STM32F405xx/07xx and STM32F415xx/17xx)

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	Reserved																														MEM_MODE	
	Reset value																															x	x
0x04	SYSCFG_PMC	Reserved								MII_RMII_SEL	Reserved							Reserved															
	Reset value									0																							
0x08	SYSCFG_EXTICR1	Reserved																EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	SYSCFG_EXTICR2	Reserved																EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SYSCFG_EXTICR3	Reserved																EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	SYSCFG_EXTICR4	Reserved																EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	SYSCFG_CMPCR	Reserved																								READY	Reserved						CMP_PD
	Reset value																									0							0

Refer to *Section 2.3: Memory map* for the register boundary addresses.

9.3 SYSCFG registers for STM32F42xxx and STM32F43xxx

9.3.1 SYSCFG memory remap register (SYSCFG_MEMRMP)

This register is used for specific configurations on memory remap:

- Three bits are used to configure the type of memory accessible at address 0x0000 0000. These bits are used to select the physical remap by software and so, bypass the BOOT pins.
- After reset these bits take the value selected by the BOOT pins. When booting from main Flash memory with BOOT pins set to 10 [(BOOT1,BOOT0) = (1,0)] this register takes the value 0x00.
- Other bits are used to swap FMC SDRAM Bank 1/2 with FMC Bank 3/4 and configure the Flash Bank 1/2 mapping

There are two possible FMC remap at address 0x0000 0000:

- FMC Bank 1 (NOR/PSRAM 1 and 2) remap:

  Only the first two regions of Bank 1 memory controller (Bank1 NOR/PSRAM 1 and NOR/PSRAM 2) can be remapped.
- FMC SDRAM Bank 1 remap.

In remap mode at address 0x0000 0000, the CPU can access the external memory via ICode bus instead of System bus which boosts up the performance.

Address offset: 0x00

Reset value: 0x0000 000X (X is the memory mode selected by the BOOT pins)

Note: Booting from NOR Flash memory or SDRAM is not allowed. The regions can only be mapped at 0x0000 0000 through software remap.

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:10 **SWP_FMC**: FMC memory mapping swap

Set and cleared by software. These bits are used to swap the FMC SDRAM Bank 1/2 and FMC Bank 3/4 (SDRAM Bank 1/2 and NAND Bank 2/PCCARD Bank) in order to enable the code execution from SDRAM Banks without a physical remapping at 0x0000 0000 address.

00: No FMC memory mapping swap

01: SDRAM banks and NAND Bank 2/PCCARD mapping are swapped. SDRAM Bank 1 and 2 are mapped at NAND Bank 2 (0x8000 0000) and PCCARD Bank (0x9000 0000) address, respectively. NAND Bank 2 and PCCARD Bank are mapped at 0xC000 0000 and 0xD000 0000, respectively.

10: Reserved

11: Reserved

Bit 9 Reserved, must be kept at reset value.

Bit 8 **FB_MODE**: Flash Bank mode selection

Set and cleared by software. This bit controls the Flash Bank 1/2 mapping.
0: Flash Bank 1 is mapped at 0x0800 0000 (and aliased at 0x0000 0000) and Flash Bank 2 is mapped at 0x0810 0000 (and aliased at 0x0010 0000)
1: Flash Bank 2 is mapped at 0x0800 0000 (and aliased at 0x0000 0000) and Flash Bank 1 is mapped at 0x0810 0000 (and aliased at 0x0010 0000)

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

Bits 2:0 **MEM_MODE:** Memory mapping selection

Set and cleared by software. This bit controls the memory internal mapping at address 0x0000 0000. After reset these bits take the value selected by the Boot pins (except for FMC).
000: Main Flash memory mapped at 0x0000 0000
001: System Flash memory mapped at 0x0000 0000
010: FMC Bank1 (NOR/PSRAM 1 and 2) mapped at 0x0000 0000
011: Embedded SRAM (SRAM1) mapped at 0x0000 0000
100: FMC/SDRAM Bank 1 mapped at 0x0000 0000
Other configurations are reserved

Note: Refer to Section 2.3: Memory map for details about the memory mapping at address 0x0000 0000.

9.3.2 SYSCFG peripheral mode configuration register (SYSCFG_PMC)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								MII_RMII_SEL	Reserved				ADCxDC2		
								rw					rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

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

Bit 23 **MII_RMII_SEL:** Ethernet PHY interface selection

Set and Cleared by software.These bits control the PHY interface for the Ethernet MAC.
0: MII interface is selected
1: RMII PHY interface is selected

Note: This configuration must be done while the MAC is under reset and before enabling the MAC clocks.

Bits 22:19 Reserved, must be kept at reset value.

Bits 18:16 **ADCxDC2**:

0: No effect.

1: Refer to AN4073 on how to use this bit.

Note: These bits can be set only if the following conditions are met:

- ADC clock higher or equal to 30 MHz.

- Only one ADCxDC2 bit must be selected if ADC conversions do not start at the same time and the sampling times differ.

- These bits must not be set when the ADCDC1 bit is set in PWR_CR register.

Bits 15:0 Reserved, must be kept at reset value.

9.3.3 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 0 to 3)

These bits are written by software to select the source input for the EXTIx external interrupt.

0000: PA[x] pin

0001: PB[x] pin

0010: PC[x] pin

0011: PD[x] pin

0100: PE[x] pin

0101: PF[x] pin

0110: PG[x] pin

0111: PH[x] pin

1000: PI[x] pin

1001: PJ[x] pin

1010: PK[x] pin

9.3.4 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 4 to 7)

These bits are written by software to select the source input for the EXTIx external interrupt.

0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin
0111: PH[x] pin
1000: PI[x] pin
1001: PJ[x] pin
1010: PK[x] pin

9.3.5 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 8 to 11)

These bits are written by software to select the source input for the EXTIx external interrupt.

0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin
0111: PH[x] pin
1000: PI[x] pin
1001: PJ[x] pin

Note: PK[11:8] are not used.

9.3.6 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 12 to 15)

These bits are written by software to select the source input for the EXTIx external interrupt.

0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin
0111: PH[x] pin
1000: PI[x] pin
1001: PJ[x] pin

Note: PK[15:12] are not used.

9.3.7 Compensation cell control register (SYSCFG_CMPCR)

Address offset: 0x20

Reset value: 0x0000 0000

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

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

Bit 8 **READY:** Compensation cell ready flag
0: I/O compensation cell not ready
1: O compensation cell ready

Bits 7:2 Reserved, must be kept at reset value.

Bit 0 **CMP_PD:** Compensation cell power-down
0: I/O compensation cell power-down mode
1: I/O compensation cell enabled

9.3.8 SYSCFG register maps for STM32F42xxx and STM32F43xxx

The following table gives the SYSCFG register map and the reset values.

Table 41. SYSCFG register map and reset values (STM32F42xxx and STM32F43xxx)

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	Reserved																				SWP_FMC		Reserved	FB_MODE	Reserved					MEM_ MODE		
	Reset value																					0	0		0						x	x	x
0x04	SYSCFG_PMC	Reserved								MII_RMII_SEL	Reserved				ADC3DC2	ADC2DC2	ADC1DC2	Reserved															
	Reset value									0					0	0	0																
0x08	SYSCFG_EXTICR1	Reserved																EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	SYSCFG_EXTICR2	Reserved																EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SYSCFG_EXTICR3	Reserved																EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	SYSCFG_EXTICR4	Reserved																EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	SYSCFG_CMPCR	Reserved																								READY	Reserved						CMP_PD
	Reset value																									0							0

Refer to *Section 2.3: Memory map* for the register boundary addresses.

10 DMA controller (DMA)

This section applies to the whole STM32F4xx family, unless otherwise specified.

10.1 DMA introduction

Direct memory access (DMA) is used in order to provide high-speed data transfer between peripherals and memory and between memory and memory. Data can be quickly moved by DMA without any CPU action. This keeps CPU resources free for other operations.

The DMA controller combines a powerful dual AHB master bus architecture with independent FIFO to optimize the bandwidth of the system, based on a complex bus matrix architecture.

The two DMA controllers have 16 streams in total (8 for each controller), each dedicated to managing memory access requests from one or more peripherals. Each stream can have up to 8 channels (requests) in total. And each has an arbiter for handling the priority between DMA requests.

10.2 DMA main features

The main DMA features are:

- Dual AHB master bus architecture, one dedicated to memory accesses and one dedicated to peripheral accesses
- AHB slave programming interface supporting only 32-bit accesses
- 8 streams for each DMA controller, up to 8 channels (requests) per stream
- Four-word depth 32 first-in, first-out memory buffers (FIFOs) per stream, that can be used in FIFO mode or direct mode:
  - FIFO mode: with threshold level software selectable between 1/4, 1/2 or 3/4 of the FIFO size
  - Direct mode

    Each DMA request immediately initiates a transfer from/to the memory. When it is configured in direct mode (FIFO disabled), to transfer data in memory-to-peripheral mode, the DMA preloads only one data from the memory to the internal

FIFO to ensure an immediate data transfer as soon as a DMA request is triggered by a peripheral.

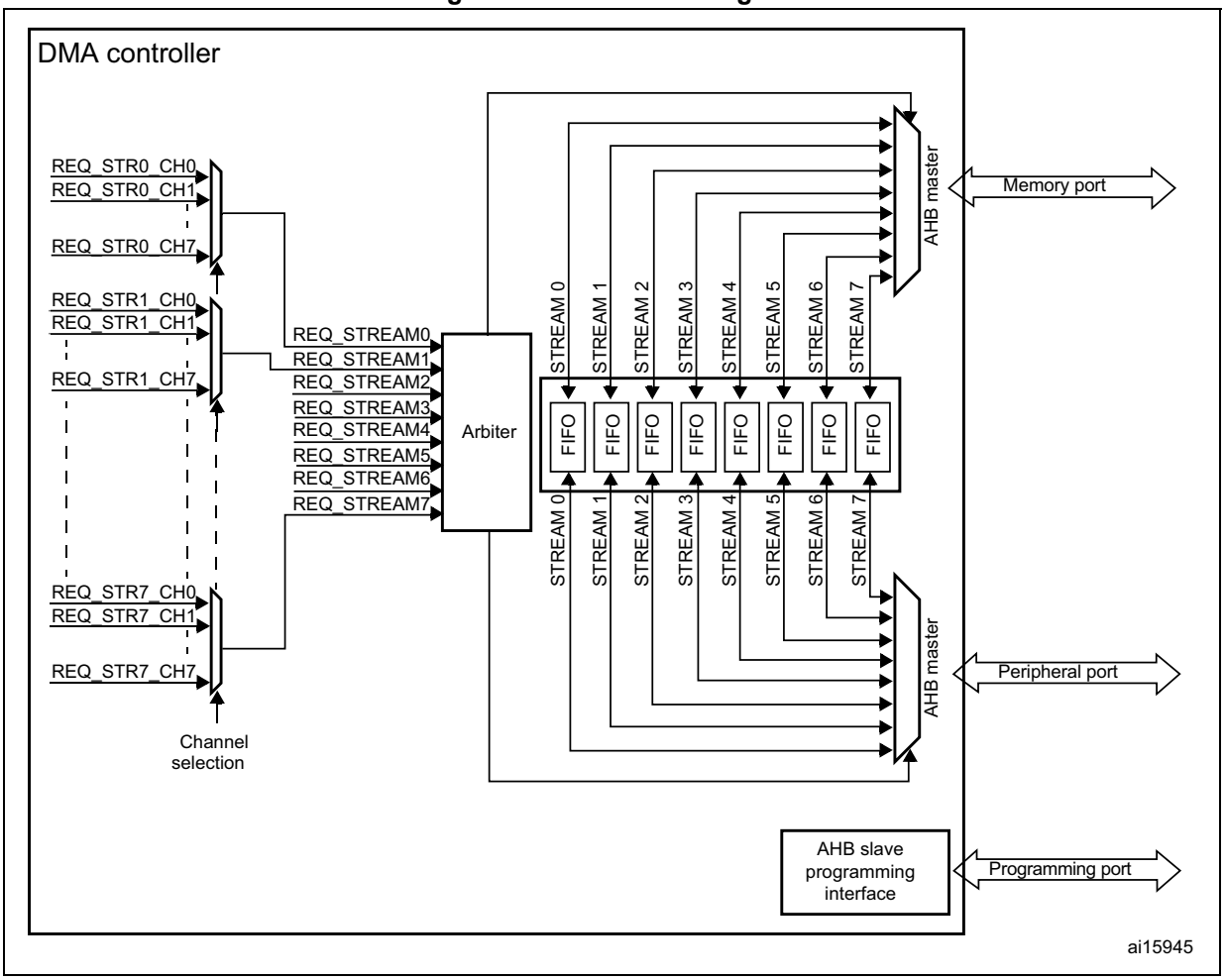
- Each stream can be configured by hardware to be:
  - a regular channel that supports peripheral-to-memory, memory-to-peripheral and memory-to-memory transfers
  - a double buffer channel that also supports double buffering on the memory side
- Each of the 8 streams are connected to dedicated hardware DMA channels (requests)
- Priorities between DMA stream requests are software-programmable (4 levels consisting of very high, high, medium, low) or hardware in case of equality (request 0 has priority over request 1, etc.)
- Each stream also supports software trigger for memory-to-memory transfers (only available for the DMA2 controller)
- Each stream request can be selected among up to 8 possible channel requests. This selection is software-configurable and allows several peripherals to initiate DMA requests
- The number of data items to be transferred can be managed either by the DMA controller or by the peripheral:
  - DMA flow controller: the number of data items to be transferred is software-programmable from 1 to 65535
  - Peripheral flow controller: the number of data items to be transferred is unknown and controlled by the source or the destination peripheral that signals the end of the transfer by hardware
- Independent source and destination transfer width (byte, half-word, word): when the data widths of the source and destination are not equal, the DMA automatically packs/unpacks the necessary transfers to optimize the bandwidth. This feature is only available in FIFO mode
- Incrementing or non-incrementing addressing for source and destination
- Supports incremental burst transfers of 4, 8 or 16 beats. The size of the burst is software-configurable, usually equal to half the FIFO size of the peripheral
- Each stream supports circular buffer management
- 5 event flags (DMA Half Transfer, DMA Transfer complete, DMA Transfer Error, DMA FIFO Error, Direct Mode Error) logically ORed together in a single interrupt request for each stream

10.3 DMA functional description

10.3.1 General description

Figure 32 shows the block diagram of a DMA.

Figure 32. DMA block diagram



The DMA controller performs direct memory transfer: as an AHB master, it can take the control of the AHB bus matrix to initiate AHB transactions.

It can carry out the following transactions:

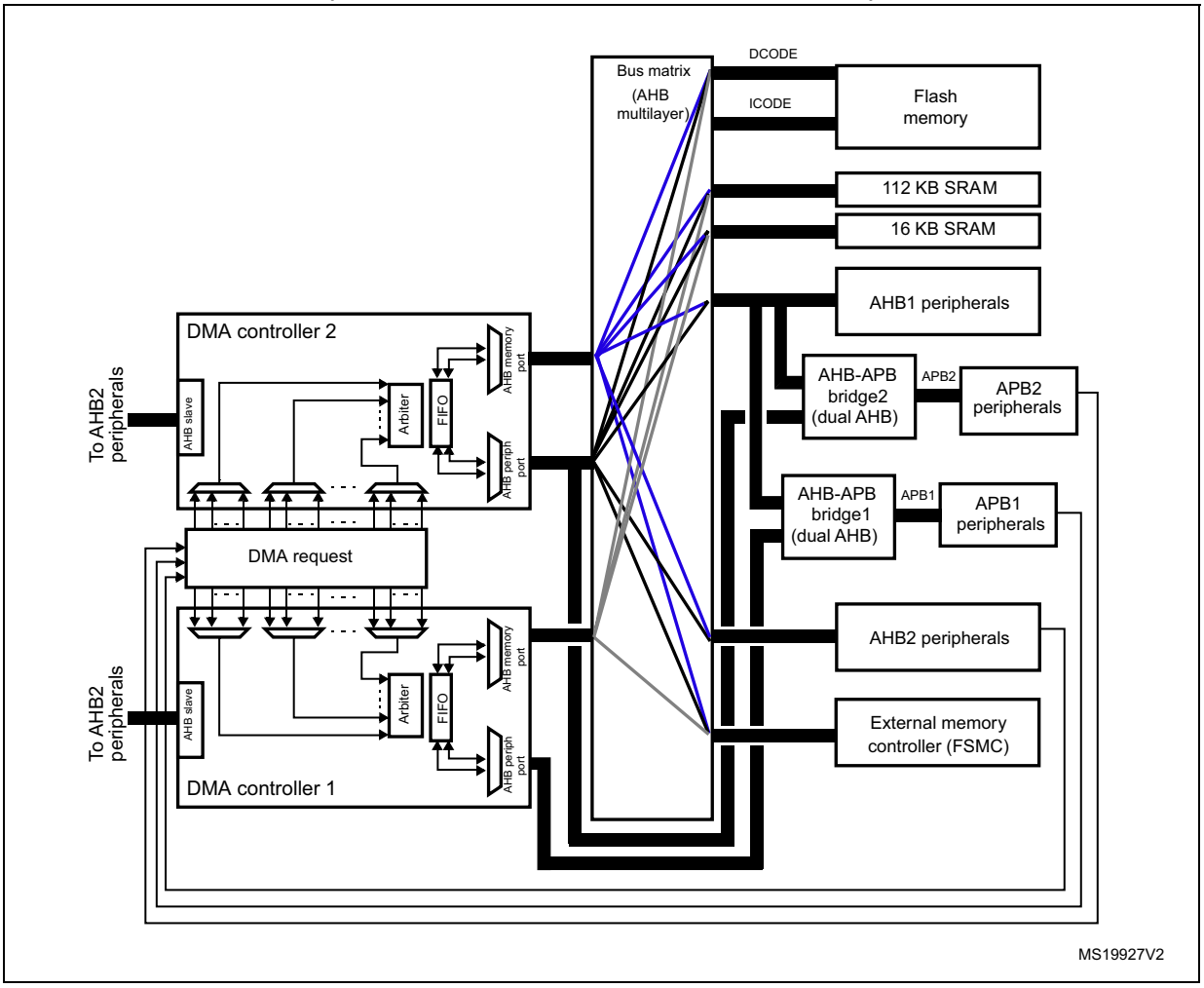
- peripheral-to-memory
- memory-to-peripheral
- memory-to-memory

The DMA controller provides two AHB master ports: the *AHB memory port*, intended to be connected to memories and the *AHB peripheral port*, intended to be connected to peripherals. However, to allow memory-to-memory transfers, the *AHB peripheral port* must also have access to the memories.

The AHB slave port is used to program the DMA controller (it supports only 32-bit accesses).

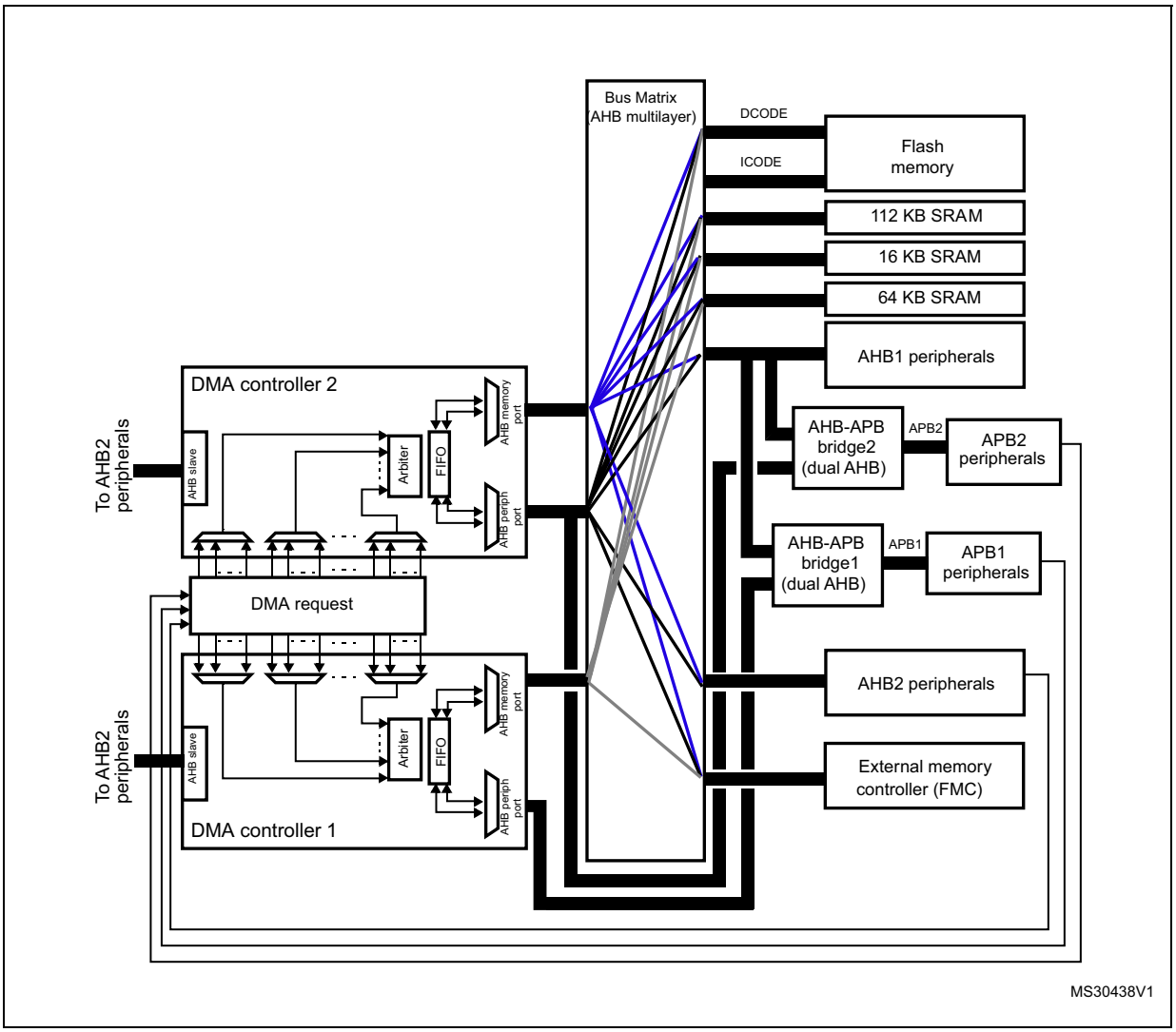
See *Figure 33* and *Figure 34* for the implementation of the system of two DMA controllers.

Figure 33. System implementation of the two DMA controllers (STM32F405xx/07xx and STM32F415xx/17xx)



1. The DMA1 controller AHB peripheral port is not connected to the bus matrix like DMA2 controller. As a result, only DMA2 streams are able to perform memory-to-memory transfers.

Figure 34. System implementation of the two DMA controllers (STM32F42xxx and STM32F43xxx)



1. The DMA1 controller AHB peripheral port is not connected to the bus matrix like in the case of the DMA2 controller, thus only DMA2 streams are able to perform memory-to-memory transfers.

10.3.2 DMA transactions

A DMA transaction consists of a sequence of a given number of data transfers. The number of data items to be transferred and their width (8-bit, 16-bit or 32-bit) are software-programmable.

Each DMA transfer consists of three operations:

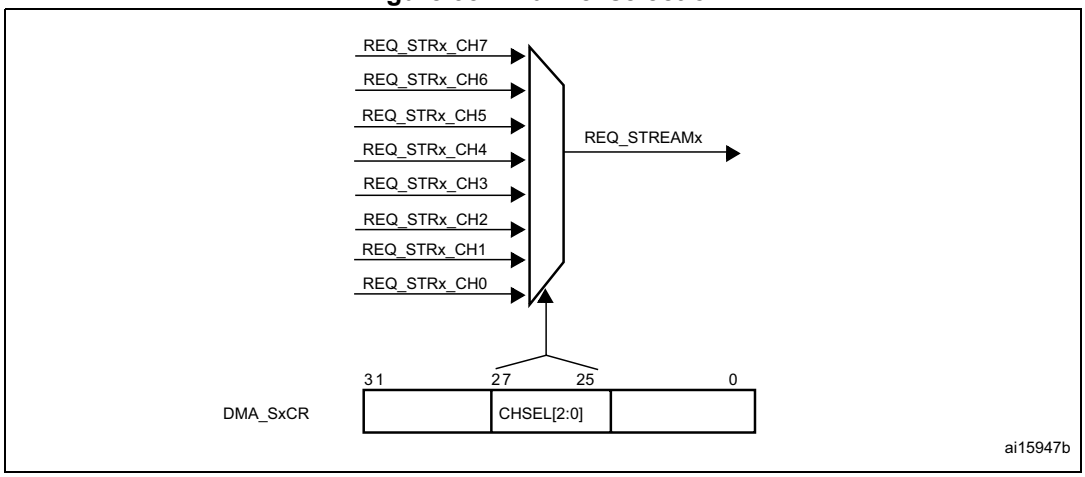
- A loading from the peripheral data register or a location in memory, addressed through the DMA_SxPAR or DMA_SxM0AR register
- A storage of the data loaded to the peripheral data register or a location in memory addressed through the DMA_SxPAR or DMA_SxM0AR register
- A post-decrement of the DMA_SxNDTR register, which contains the number of transactions that still have to be performed

After an event, the peripheral sends a request signal to the DMA controller. The DMA controller serves the request depending on the channel priorities. As soon as the DMA controller accesses the peripheral, an Acknowledge signal is sent to the peripheral by the DMA controller. The peripheral releases its request as soon as it gets the Acknowledge signal from the DMA controller. Once the request has been deasserted by the peripheral, the DMA controller releases the Acknowledge signal. If there are more requests, the peripheral can initiate the next transaction.

10.3.3 Channel selection

Each stream is associated with a DMA request that can be selected out of 8 possible channel requests. The selection is controlled by the CHSEL[2:0] bits in the DMA_SxCR register.

Figure 35. Channel selection



The 8 requests from the peripherals (TIM, ADC, SPI, I2C, etc.) are independently connected to each channel and their connection depends on the product implementation.

See the following table(s) for examples of DMA request mappings.

Table 42. DMA1 request mapping

Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 0	SPI3_RX	-	SPI3_RX	SPI2_RX	SPI2_TX	SPI3_TX	-	SPI3_TX
Channel 1	I2C1_RX	-	TIM7_UP	-	TIM7_UP	I2C1_RX	I2C1_TX	I2C1_TX
Channel 2	TIM4_CH1	-	I2S3_EXT_RX	TIM4_CH2	I2S2_EXT_TX	I2S3_EXT_TX	TIM4_UP	TIM4_CH3
Channel 3	I2S3_EXT_RX	TIM2_UP TIM2_CH3	I2C3_RX	I2S2_EXT_RX	I2C3_TX	TIM2_CH1	TIM2_CH2 TIM2_CH4	TIM2_UP TIM2_CH4
Channel 4	UART5_RX	USART3_RX	UART4_RX	USART3_TX	UART4_TX	USART2_RX	USART2_TX	UART5_TX
Channel 5	UART8_TX(1)	UART7_TX(1)	TIM3_CH4 TIM3_UP	UART7_RX(1)	TIM3_CH1 TIM3_TRIG	TIM3_CH2	UART8_RX(1)	TIM3_CH3

Table 42. DMA1 request mapping (continued)

Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 6	TIM5_CH3 TIM5_UP	TIM5_CH4 TIM5_TRIG	TIM5_CH1	TIM5_CH4 TIM5_TRIG	TIM5_CH2	-	TIM5_UP	-
Channel 7	-	TIM6_UP	I2C2_RX	I2C2_RX	USART3_TX	DAC1	DAC2	I2C2_TX

1. These requests are available on STM32F42xxx and STM32F43xxx only.

Table 43. DMA2 request mapping

Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 0	ADC1	SAI1_A(1)	TIM8_CH1 TIM8_CH2 TIM8_CH3	SAI1_A(1)	ADC1	SAI1_B(1)	TIM1_CH1 TIM1_CH2 TIM1_CH3	-
Channel 1	-	DCMI	ADC2	ADC2	SAI1_B(1)	SPI6_TX(1)	SPI6_RX(1)	DCMI
Channel 2	ADC3	ADC3	-	SPI5_RX(1)	SPI5_TX(1)	CRYP_OUT	CRYP_IN	HASH_IN
Channel 3	SPI1_RX	-	SPI1_RX	SPI1_TX	-	SPI1_TX	-	-
Channel 4	SPI4_RX(1)	SPI4_TX(1)	USART1_RX	SDIO	-	USART1_RX	SDIO	USART1_TX
Channel 5	-	USART6_RX	USART6_RX	SPI4_RX(1)	SPI4_TX(1)	-	USART6_TX	USART6_TX
Channel 6	TIM1_TRIG	TIM1_CH1	TIM1_CH2	TIM1_CH1	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	-
Channel 7	-	TIM8_UP	TIM8_CH1	TIM8_CH2	TIM8_CH3	SPI5_RX(1)	SPI5_TX(1)	TIM8_CH4 TIM8_TRIG TIM8_COM

1. These requests are available on STM32F42xxx and STM32F43xxx.

10.3.4 Arbiter

An arbiter manages the 8 DMA stream requests based on their priority for each of the two AHB master ports (memory and peripheral ports) and launches the peripheral/memory access sequences.

Priorities are managed in two stages:

- Software: each stream priority can be configured in the DMA_SxCR register. There are four levels:
  - Very high priority
  - High priority
  - Medium priority
  - Low priority
- Hardware: If two requests have the same software priority level, the stream with the lower number takes priority over the stream with the higher number. For example, Stream 2 takes priority over Stream 4.

10.3.5 DMA streams

Each of the 8 DMA controller streams provides a unidirectional transfer link between a source and a destination.

Each stream can be configured to perform:

- Regular type transactions: memory-to-peripherals, peripherals-to-memory or memory-to-memory transfers
- Double-buffer type transactions: double buffer transfers using two memory pointers for the memory (while the DMA is reading/writing from/to a buffer, the application can write/read to/from the other buffer).

The amount of data to be transferred (up to 65535) is programmable and related to the source width of the peripheral that requests the DMA transfer connected to the peripheral AHB port. The register that contains the amount of data items to be transferred is decremented after each transaction.

10.3.6 Source, destination and transfer modes

Both source and destination transfers can address peripherals and memories in the entire 4 GB area, at addresses comprised between 0x0000 0000 and 0xFFFF FFFF.

The direction is configured using the DIR[1:0] bits in the DMA_SxCR register and offers three possibilities: memory-to-peripheral, peripheral-to-memory or memory-to-memory transfers. *Table 44* describes the corresponding source and destination addresses.

Table 44. Source and destination address

Bits DIR[1:0] of the DMA_SxCR register	Direction	Source address	Destination address
00	Peripheral-to-memory	DMA_SxPAR	DMA_SxM0AR
01	Memory-to-peripheral	DMA_SxM0AR	DMA_SxPAR
10	Memory-to-memory	DMA_SxPAR	DMA_SxM0AR
11	reserved	-	-

When the data width (programmed in the PSIZE or MSIZE bits in the DMA_SxCR register) is a half-word or a word, respectively, the peripheral or memory address written into the DMA_SxPAR or DMA_SxM0AR/M1AR registers has to be aligned on a word or half-word address boundary, respectively.

Peripheral-to-memory mode

Figure 36 describes this mode.

When this mode is enabled (by setting the bit EN in the DMA_SxCR register), each time a peripheral request occurs, the stream initiates a transfer from the source to fill the FIFO.

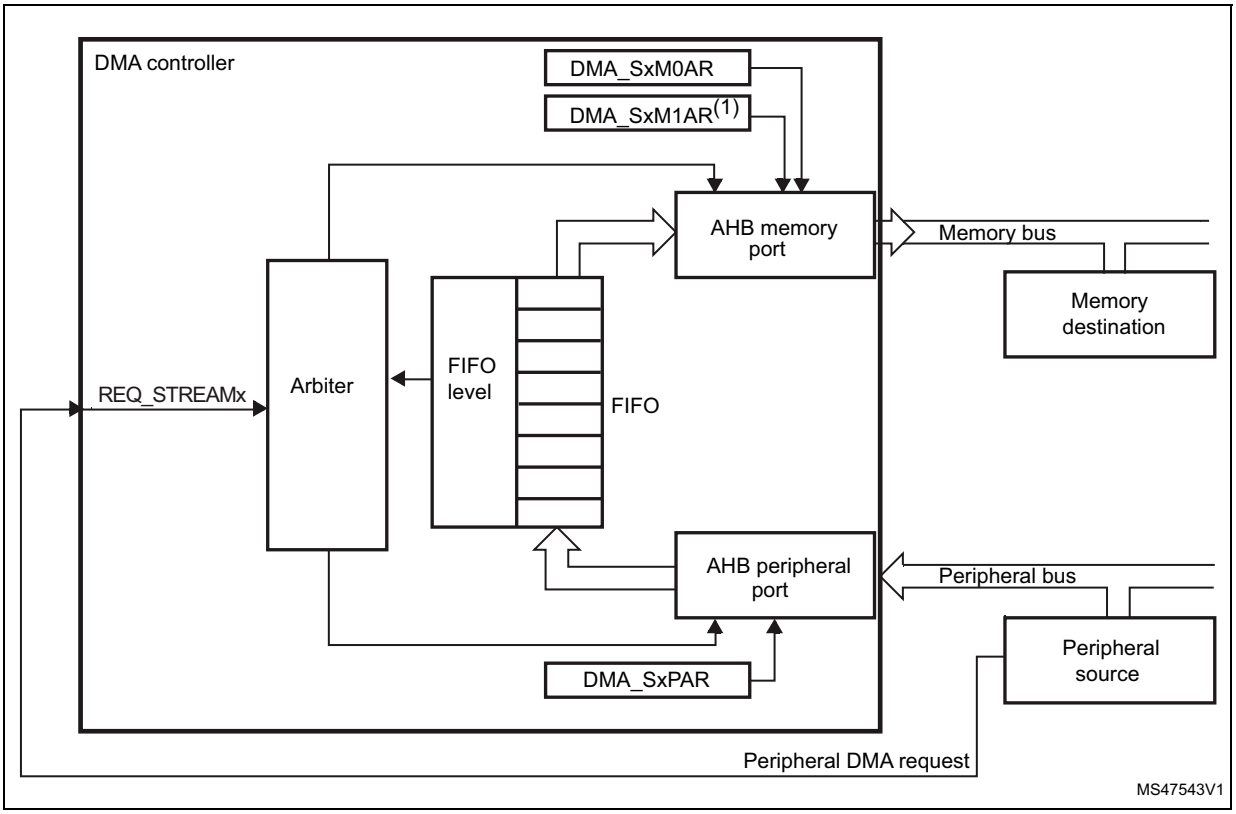
When the threshold level of the FIFO is reached, the contents of the FIFO are drained and stored into the destination.

The transfer stops once the DMA_SxNDTR register reaches zero, when the peripheral requests the end of transfers (in case of a peripheral flow controller) or when the EN bit in the DMA_SxCR register is cleared by software.

In direct mode (when the DMDIS value in the DMA_SxFCR register is ‘0’), the threshold level of the FIFO is not used: after each single data transfer from the peripheral to the FIFO, the corresponding data are immediately drained and stored into the destination.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.

Figure 36. Peripheral-to-memory mode



1. For double-buffer mode.

Memory-to-peripheral mode

Figure 37 describes this mode.

When this mode is enabled (by setting the EN bit in the DMA_SxCR register), the stream immediately initiates transfers from the source to entirely fill the FIFO.

Each time a peripheral request occurs, the contents of the FIFO are drained and stored into the destination. When the level of the FIFO is lower than or equal to the predefined threshold level, the FIFO is fully reloaded with data from the memory.

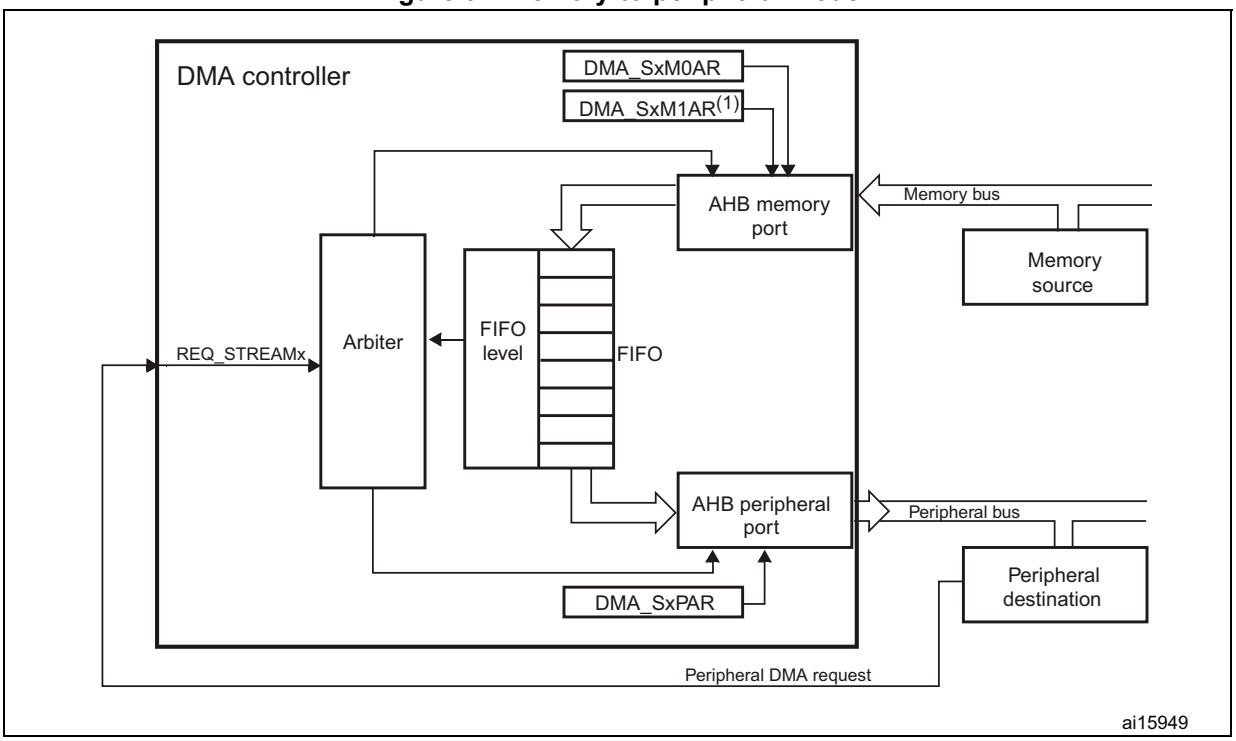
The transfer stops once the DMA_SxNDTR register reaches zero, when the peripheral requests the end of transfers (in case of a peripheral flow controller) or when the EN bit in the DMA_SxCR register is cleared by software.

In direct mode (when the DMDIS value in the DMA_SxFCR register is '0'), the threshold level of the FIFO is not used. Once the stream is enabled, the DMA preloads the first data to transfer into an internal FIFO. As soon as the peripheral requests a data transfer, the DMA transfers the preloaded value into the configured destination. It then reloads again the

empty internal FIFO with the next data to be transfer. The preloaded data size corresponds to the value of the PSIZE bitfield in the DMA_SxCR register.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.

Figure 37. Memory-to-peripheral mode



1. For double-buffer mode.

Memory-to-memory mode

The DMA channels can also work without being triggered by a request from a peripheral. This is the memory-to-memory mode, described in *Figure 38*.

When the stream is enabled by setting the Enable bit (EN) in the DMA_SxCR register, the stream immediately starts to fill the FIFO up to the threshold level. When the threshold level is reached, the FIFO contents are drained and stored into the destination.

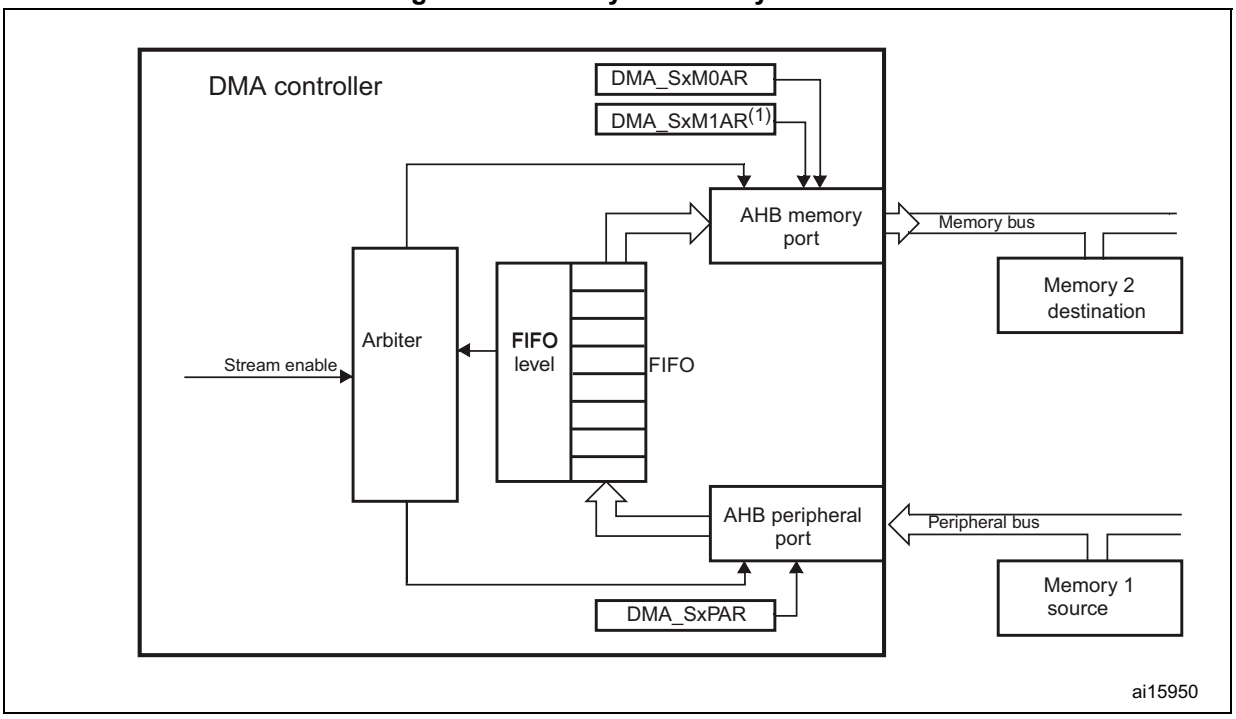
The transfer stops once the DMA_SxNDTR register reaches zero or when the EN bit in the DMA_SxCR register is cleared by software.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.

Note: When memory-to-memory mode is used, the Circular and direct modes are not allowed.

Only the DMA2 controller is able to perform memory-to-memory transfers.

Figure 38. Memory-to-memory mode



1. For double-buffer mode.

10.3.7 Pointer incrementation

Peripheral and memory pointers can optionally be automatically post-incremented or kept constant after each transfer depending on the PINC and MINC bits in the DMA_SxCR register.

Disabling the Increment mode is useful when the peripheral source or destination data are accessed through a single register.

If the Increment mode is enabled, the address of the next transfer will be the address of the previous one incremented by 1 (for bytes), 2 (for half-words) or 4 (for words) depending on the data width programmed in the PSIZE or MSIZE bits in the DMA_SxCR register.

In order to optimize the packing operation, it is possible to fix the increment offset size for the peripheral address whatever the size of the data transferred on the AHB peripheral port. The PINCOS bit in the DMA_SxCR register is used to align the increment offset size with the data size on the peripheral AHB port, or on a 32-bit address (the address is then incremented by 4). The PINCOS bit has an impact on the AHB peripheral port only.

If PINCOS bit is set, the address of the next transfer is the address of the previous one incremented by 4 (automatically aligned on a 32-bit address) whatever the PSIZE value. The AHB memory port, however, is not impacted by this operation.

10.3.8 Circular mode

The Circular mode is available to handle circular buffers and continuous data flows (e.g. ADC scan mode). This feature can be enabled using the CIRC bit in the DMA_SxCR register.

When the circular mode is activated, the number of data items to be transferred is automatically reloaded with the initial value programmed during the stream configuration phase, and the DMA requests continue to be served.

Note: In the circular mode, it is mandatory to respect the following rule in case of a burst mode configured for memory:

DMA_SxNDTR = Multiple of ((Mburst beat) × (Msize)/(Psize)), where:

- *(Mburst beat) = 4, 8 or 16 (depending on the MBURST bits in the DMA_SxCR register)*
- *((Msize)/(Psize)) = 1, 2, 4, 1/2 or 1/4 (Msize and Psize represent the MSIZE and PSIZE bits in the DMA_SxCR register. They are byte dependent)*
- *DMA_SxNDTR = Number of data items to transfer on the AHB peripheral port*

For example: Mburst beat = 8 (INCR8), MSIZE = '00' (byte) and PSIZE = '01' (half-word), in this case: DMA_SxNDTR must be a multiple of (8 × 1/2 = 4).

If this formula is not respected, the DMA behavior and data integrity are not guaranteed.

NDTR must also be a multiple of the Peripheral burst size multiplied by the peripheral data size, otherwise this could result in a bad DMA behavior.

10.3.9 Double buffer mode

This mode is available for all the DMA1 and DMA2 streams.

The Double buffer mode is enabled by setting the DBM bit in the DMA_SxCR register.

A double-buffer stream works as a regular (single buffer) stream with the difference that it has two memory pointers. When the Double buffer mode is enabled, the Circular mode is automatically enabled (CIRC bit in DMA_SxCR is don't care) and at each end of transaction, the memory pointers are swapped.

In this mode, the DMA controller swaps from one memory target to another at each end of transaction. This allows the software to process one memory area while the second memory area is being filled/used by the DMA transfer. The double-buffer stream can work in both directions (the memory can be either the source or the destination) as described in *Table 45: Source and destination address registers in Double buffer mode (DBM=1)*.

*Note: In Double buffer mode, it is possible to update the base address for the AHB memory port on-the-fly (*DMA_SxM0AR or DMA_SxM1AR*) when the stream is enabled, by respecting the following conditions:*

- *When the CT bit is '0' in the DMA_SxCR register, the DMA_SxM1AR register can be written. Attempting to write to this register while CT = '1' sets an error flag (TEIF) and the stream is automatically disabled.*
- *When the CT bit is '1' in the DMA_SxCR register, the DMA_SxM0AR register can be written. Attempting to write to this register while CT = '0', sets an error flag (TEIF) and the stream is automatically disabled.*

To avoid any error condition, it is advised to change the base address as soon as the TCIF flag is asserted because, at this point, the targeted memory must have changed from

memory 0 to 1 (or from 1 to 0) depending on the value of CT in the DMA_SxCR register in accordance with one of the two above conditions.

For all the other modes (except the Double buffer mode), the memory address registers are write-protected as soon as the stream is enabled.

Table 45. Source and destination address registers in Double buffer mode (DBM=1)

Bits DIR[1:0] of the DMA_SxCR register	Direction	Source address	Destination address
00	Peripheral-to-memory	DMA_SxPAR	DMA_SxM0AR / DMA_SxM1AR
01	Memory-to-peripheral	DMA_SxM0AR / DMA_SxM1AR	DMA_SxPAR
10	Not allowed(1)		
11	Reserved	-	-

1. When the Double buffer mode is enabled, the Circular mode is automatically enabled. Since the memory-to-memory mode is not compatible with the Circular mode, when the Double buffer mode is enabled, it is not allowed to configure the memory-to-memory mode.

10.3.10 Programmable data width, packing/unpacking, endianess

The number of data items to be transferred has to be programmed into DMA_SxNDTR (number of data items to transfer bit, NDT) before enabling the stream (except when the flow controller is the peripheral, PFCTRL bit in DMA_SxCR is set).

When using the internal FIFO, the data widths of the source and destination data are programmable through the PSIZE and MSIZE bits in the DMA_SxCR register (can be 8-, 16- or 32-bit).

When PSIZE and MSIZE are not equal:

- The data width of the number of data items to transfer, configured in the DMA_SxNDTR register is equal to the width of the peripheral bus (configured by the PSIZE bits in the DMA_SxCR register). For instance, in case of peripheral-to-memory, memory-to-peripheral or memory-to-memory transfers and if the PSIZE[1:0] bits are configured for half-word, the number of bytes to be transferred is equal to 2 × NDT.
- The DMA controller only copes with little-endian addressing for both source and destination. This is described in *Table 46: Packing/unpacking & endian behavior (bit PINC = MINC = 1)*.

This packing/unpacking procedure may present a risk of data corruption when the operation is interrupted before the data are completely packed/unpacked. So, to ensure data coherence, the stream may be configured to generate burst transfers: in this case, each group of transfers belonging to a burst are indivisible (refer to *Section 10.3.11: Single and burst transfers*).

In direct mode (DMDIS = 0 in the DMA_SxFCR register), the packing/unpacking of data is not possible. In this case, it is not allowed to have different source and destination transfer data widths: both are equal and defined by the PSIZE bits in the DMA_SxCR MSIZE bits are don't care).

Table 46. Packing/unpacking & endian behavior (bit PINC = MINC = 1)

AHB memory port width	AHB peripheral port width	Number of data items to transfer (NDT)	Memory transfer number	Memory port address / byte lane	Peripheral transfer number	Peripheral port address / byte lane	
						PINCOS = 1	PINCOS = 0
8	8	4	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
8	16	2	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1 2	0x0 / B1|B0[15:0] 0x4 / B3|B2[15:0]	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]
8	32	1	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1	0x0 / B3|B2|B1|B0[31:0]	0x0 / B3|B2|B1|B0[31:0]
16	8	4	1 2	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
16	16	2	1 2	0x0 / B1|B0[15:0] 0x2 / B1|B0[15:0]	1 2	0x0 / B1|B0[15:0] 0x4 / B3|B2[15:0]	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]
16	32	1	1 2	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]	1	0x0 / B3|B2|B1|B0[31:0]	0x0 / B3|B2|B1|B0[31:0]
32	8	4	1	0x0 / B3|B2|B1|B0[31:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
32	16	2	1	0x0 /B3|B2|B1|B0[31:0]	1 2	0x0 / B1|B0[15:0] 0x4 / B3|B2[15:0]	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]
32	32	1	1	0x0 /B3|B2|B1|B0 [31:0]	1	0x0 /B3|B2|B1|B0 [31:0]	0x0 / B3|B2|B1|B0[31:0]

Note: Peripheral port may be the source or the destination (it could also be the memory source in the case of memory-to-memory transfer).

PSIZE, MSIZE and NDT[15:0] have to be configured so as to ensure that the last transfer will not be incomplete. This can occur when the data width of the peripheral port (PSIZE bits) is lower than the data width of the memory port (MSIZE bits). This constraint is summarized in *Table 47*.

Table 47. Restriction on NDT versus PSIZE and MSIZE

PSIZE[1:0] of DMA_SxCR	MSIZE[1:0] of DMA_SxCR	NDT[15:0] of DMA_SxNDTR
00 (8-bit)	01 (16-bit)	must be a multiple of 2
00 (8-bit)	10 (32-bit)	must be a multiple of 4
01 (16-bit)	10 (32-bit)	must be a multiple of 2

10.3.11 Single and burst transfers

The DMA controller can generate single transfers or incremental burst transfers of 4, 8 or 16 beats.

The size of the burst is configured by software independently for the two AHB ports by using the MBURST[1:0] and PBURST[1:0] bits in the DMA_SxCR register.

The burst size indicates the number of beats in the burst, not the number of bytes transferred.

To ensure data coherence, each group of transfers that form a burst are indivisible: AHB transfers are locked and the arbiter of the AHB bus matrix does not degrant the DMA master during the sequence of the burst transfer.

Depending on the single or burst configuration, each DMA request initiates a different number of transfers on the AHB peripheral port:

- When the AHB peripheral port is configured for single transfers, each DMA request generates a data transfer of a byte, half-word or word depending on the PSIZE[1:0] bits in the DMA_SxCR register
- When the AHB peripheral port is configured for burst transfers, each DMA request generates 4,8 or 16 beats of byte, half word or word transfers depending on the PBURST[1:0] and PSIZE[1:0] bits in the DMA_SxCR register.

The same as above has to be considered for the AHB memory port considering the MBURST and MSIZE bits.

In direct mode, the stream can only generate single transfers and the MBURST[1:0] and PBURST[1:0] bits are forced by hardware.

The address pointers (DMA_SxPAR or DMA_SxM0AR registers) must be chosen so as to ensure that all transfers within a burst block are aligned on the address boundary equal to the size of the transfer.

The burst configuration has to be selected in order to respect the AHB protocol, where bursts must *not* cross the 1 KB address boundary because the minimum address space that can be allocated to a single slave is 1 KB. This means that the 1 KB address boundary should not be crossed by a burst block transfer, otherwise an AHB error would be generated, that is not reported by the DMA registers.

10.3.12 FIFO

FIFO structure

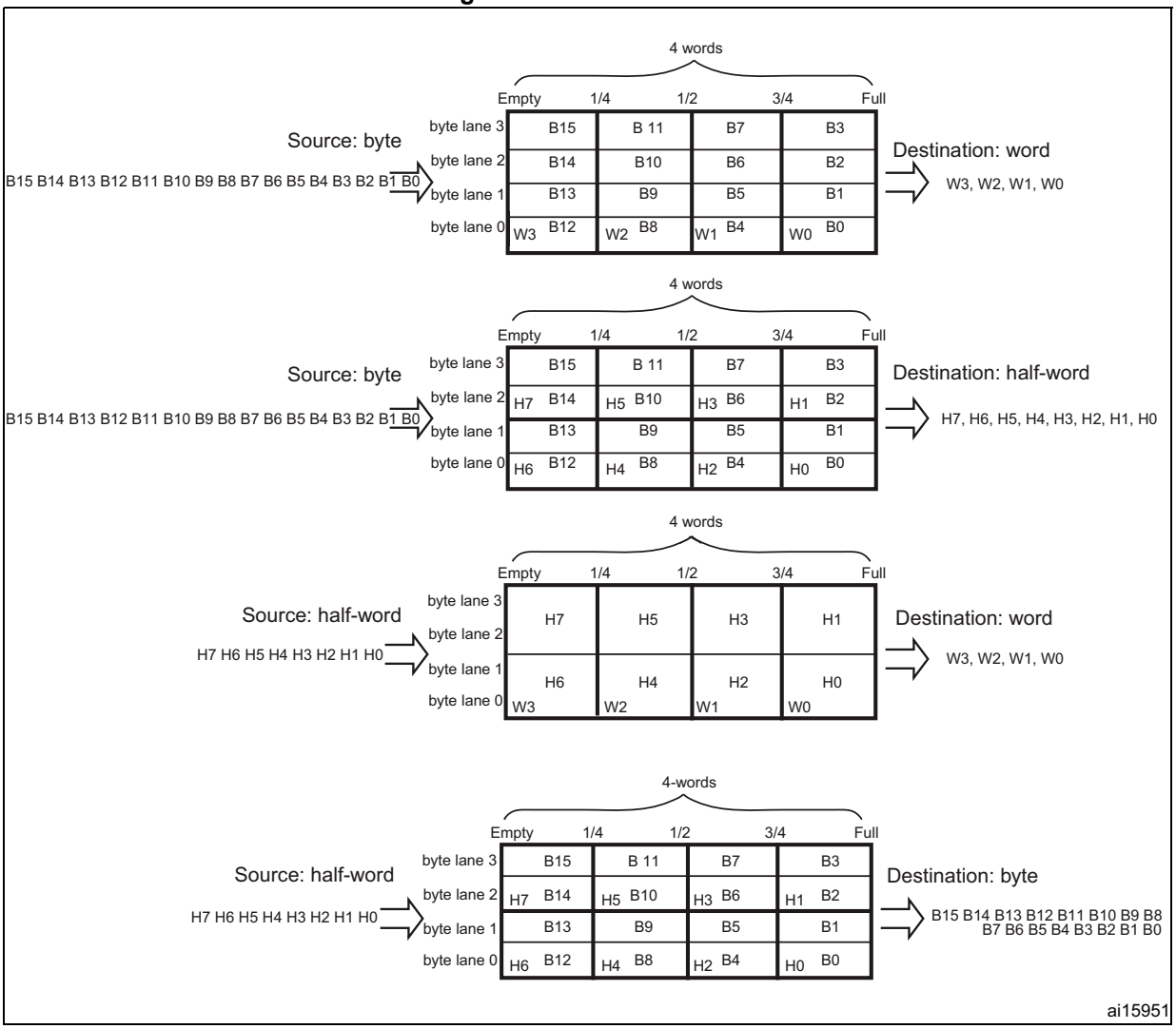
The FIFO is used to temporarily store data coming from the source before transmitting them to the destination.

Each stream has an independent 4-word FIFO and the threshold level is software-configurable between 1/4, 1/2, 3/4 or full.

To enable the use of the FIFO threshold level, the direct mode must be disabled by setting the DMDIS bit in the DMA_SxFCR register.

The structure of the FIFO differs depending on the source and destination data widths, and is described in *Figure 39: FIFO structure*.

Figure 39. FIFO structure



FIFO threshold and burst configuration

Caution is required when choosing the FIFO threshold (bits FTH[1:0] of the DMA_SxFCR register) and the size of the memory burst (MBURST[1:0] of the DMA_SxCR register): The content pointed by the FIFO threshold must exactly match to an integer number of memory burst transfers. If this is not in the case, a FIFO error (flag FEIFx of the DMA_HISR or DMA_LISR register) will be generated when the stream is enabled, then the stream will be automatically disabled. The allowed and forbidden configurations are described in the *Table 48: FIFO threshold configurations*.

Table 48. FIFO threshold configurations

MSIZE	FIFO level	MBURST = INCR4	MBURST = INCR8	MBURST = INCR16
Byte	1/4	1 burst of 4 beats	forbidden	forbidden
	1/2	2 bursts of 4 beats	1 burst of 8 beats	
	3/4	3 bursts of 4 beats	forbidden	
	Full	4 bursts of 4 beats	2 bursts of 8 beats	1 burst of 16 beats
Half-word	1/4	forbidden	forbidden	forbidden
	1/2	1 burst of 4 beats		
	3/4	forbidden		
	Full	2 bursts of 4 beats	1 burst of 8 beats	
Word	1/4	forbidden	forbidden	
	1/2			
	3/4			
	Full	1 burst of 4 beats		

In all cases, the burst size multiplied by the data size must not exceed the FIFO size (data size can be: 1 (byte), 2 (half-word) or 4 (word)).

Incomplete Burst transfer at the end of a DMA transfer may happen if one of the following conditions occurs:

- For the AHB peripheral port configuration: the total number of data items (set in the DMA_SxNDTR register) is not a multiple of the burst size multiplied by the data size
- For the AHB memory port configuration: the number of remaining data items in the FIFO to be transferred to the memory is not a multiple of the burst size multiplied by the data size

In such cases, the remaining data to be transferred will be managed in single mode by the DMA, even if a burst transaction was requested during the DMA stream configuration.

Note: When burst transfers are requested on the peripheral AHB port and the FIFO is used (DMDIS = 1 in the DMA_SxCR register), it is mandatory to respect the following rule to avoid permanent underrun or overrun conditions, depending on the DMA stream direction:

If (PBURST × PSIZE) = FIFO_SIZE (4 words), FIFO_Threshold = 3/4 is forbidden with PSIZE = 1, 2 or 4 and PBURST = 4, 8 or 16.

This rule ensures that enough FIFO space at a time will be free to serve the request from the peripheral.

FIFO flush

The FIFO can be flushed when the stream is disabled by resetting the EN bit in the DMA_SxCR register and when the stream is configured to manage peripheral-to-memory or memory-to-memory transfers: If some data are still present in the FIFO when the stream is disabled, the DMA controller continues transferring the remaining data to the destination (even though stream is effectively disabled). When this flush is completed, the transfer complete status bit (TCIFx) in the DMA_LISR or DMA_HISR register is set.

The remaining data counter DMA_SxNDTR keeps the value in this case to indicate how many data items are currently available in the destination memory.

Note that during the FIFO flush operation, if the number of remaining data items in the FIFO to be transferred to memory (in bytes) is less than the memory data width (for example 2 bytes in FIFO while MSIZE is configured to word), data will be sent with the data width set in the MSIZE bit in the DMA_SxCR register. This means that memory will be written with an undesired value. The software may read the DMA_SxNDTR register to determine the memory area that contains the good data (start address and last address).

If the number of remaining data items in the FIFO is lower than a burst size (if the MBURST bits in DMA_SxCR register are set to configure the stream to manage burst on the AHB memory port), single transactions will be generated to complete the FIFO flush.

Direct mode

By default, the FIFO operates in direct mode (DMDIS bit in the DMA_SxFCR is reset) and the FIFO threshold level is not used. This mode is useful when the system requires an immediate and single transfer to or from the memory after each DMA request.

When the DMA is configured in direct mode (FIFO disabled), to transfer data in memory-to-peripheral mode, the DMA preloads one data from the memory to the internal FIFO to ensure an immediate data transfer as soon as a DMA request is triggered by a peripheral.

To avoid saturating the FIFO, it is recommended to configure the corresponding stream with a high priority.

This mode is restricted to transfers where:

- The source and destination transfer widths are equal and both defined by the PSIZE[1:0] bits in DMA_SxCR (MSIZE[1:0] bits are don't care)
- Burst transfers are not possible (PBURST[1:0] and MBURST[1:0] bits in DMA_SxCR are don't care)

Direct mode must not be used when implementing memory-to-memory transfers.

10.3.13 DMA transfer completion

Different events can generate an end of transfer by setting the TCIFx bit in the DMA_LISR or DMA_HISR status register:

- In DMA flow controller mode:
  - The DMA_SxNDTR counter has reached zero in the memory-to-peripheral mode
  - The stream is disabled before the end of transfer (by clearing the EN bit in the DMA_SxCR register) and (when transfers are peripheral-to-memory or memory-

to-memory) all the remaining data have been flushed from the FIFO into the memory

- In Peripheral flow controller mode:
  - The last external burst or single request has been generated from the peripheral and (when the DMA is operating in peripheral-to-memory mode) the remaining data have been transferred from the FIFO into the memory
  - The stream is disabled by software, and (when the DMA is operating in peripheral-to-memory mode) the remaining data have been transferred from the FIFO into the memory

Note: The transfer completion is dependent on the remaining data in FIFO to be transferred into memory only in the case of peripheral-to-memory mode. This condition is not applicable in memory-to-peripheral mode.

If the stream is configured in noncircular mode, after the end of the transfer (that is when the number of data to be transferred reaches zero), the DMA is stopped (EN bit in DMA_SxCR register is cleared by Hardware) and no DMA request is served unless the software reprograms the stream and re-enables it (by setting the EN bit in the DMA_SxCR register).

10.3.14 DMA transfer suspension

At any time, a DMA transfer can be suspended to be restarted later on or to be definitively disabled before the end of the DMA transfer.

There are two cases:

- The stream disables the transfer with no later-on restart from the point where it was stopped. There is no particular action to do, except to clear the EN bit in the DMA_SxCR register to disable the stream. The stream may take time to be disabled (ongoing transfer is completed first). The transfer complete interrupt flag (TCIF in the DMA_LISR or DMA_HISR register) is set in order to indicate the end of transfer. The value of the EN bit in DMA_SxCR is now ‘0’ to confirm the stream interruption. The DMA_SxNDTR register contains the number of remaining data items at the moment when the stream was stopped so that the software can determine how many data items have been transferred before the stream was interrupted.
- The stream suspends the transfer before the number of remaining data items to be transferred in the DMA_SxNDTR register reaches 0. The aim is to restart the transfer later by re-enabling the stream. In order to restart from the point where the transfer was stopped, the software has to read the DMA_SxNDTR register after disabling the stream by writing the EN bit in DMA_SxCR register (and then checking that it is at ‘0’) to know the number of data items already collected. Then:
  - The peripheral and/or memory addresses have to be updated in order to adjust the address pointers
  - The SxNDTR register has to be updated with the remaining number of data items to be transferred (the value read when the stream was disabled)
  - The stream may then be re-enabled to restart the transfer from the point it was stopped

Note: Note that a Transfer complete interrupt flag (TCIF in DMA_LISR or DMA_HISR) is set to indicate the end of transfer due to the stream interruption.

10.3.15 Flow controller

The entity that controls the number of data to be transferred is known as the flow controller. This flow controller is configured independently for each stream using the PFCTRL bit in the DMA_SxCR register.

The flow controller can be:

- The DMA controller: in this case, the number of data items to be transferred is programmed by software into the DMA_SxNDTR register before the DMA stream is enabled.
- The peripheral source or destination: this is the case when the number of data items to be transferred is unknown. The peripheral indicates by hardware to the DMA controller when the last data are being transferred. This feature is only supported for peripherals which are able to signal the end of the transfer, that is:
  – SDIO

When the peripheral flow controller is used for a given stream, the value written into the DMA_SxNDTR has no effect on the DMA transfer. Actually, whatever the value written, it will be forced by hardware to 0xFFFF as soon as the stream is enabled, to respect the following schemes:

- Anticipated stream interruption: EN bit in DMA_SxCR register is reset to 0 by the software to stop the stream before the last data hardware signal (single or burst) is sent by the peripheral. In such a case, the stream is switched off and the FIFO flush is triggered in the case of a peripheral-to-memory DMA transfer. The TCIFx flag of the corresponding stream is set in the status register to indicate the DMA completion. To know the number of data items transferred during the DMA transfer, read the DMA_SxNDTR register and apply the following formula:
  – Number_of_data_transferred = 0xFFFF – DMA_SxNDTR
- Normal stream interruption due to the reception of a last data hardware signal: the stream is automatically interrupted when the peripheral requests the last transfer (single or burst) and when this transfer is complete. the TCIFx flag of the corresponding stream is set in the status register to indicate the DMA transfer completion. To know the number of data items transferred, read the DMA_SxNDTR register and apply the same formula as above.
- The DMA_SxNDTR register reaches 0: the TCIFx flag of the corresponding stream is set in the status register to indicate the forced DMA transfer completion. The stream is automatically switched off even though the last data hardware signal (single or burst) has not been yet asserted. The already transferred data will not be lost. This means that a maximum of 65535 data items can be managed by the DMA in a single transaction, even in peripheral flow control mode.

Note: When configured in memory-to-memory mode, the DMA is always the flow controller and the PFCTRL bit is forced to 0 by hardware.

The Circular mode is forbidden in the peripheral flow controller mode.

10.3.16 Summary of the possible DMA configurations

Table 49 summarizes the different possible DMA configurations.

Table 49. Possible DMA configurations

DMA transfer mode	Source	Destination	Flow controller	Circular mode	Transfer type	Direct mode	Double buffer mode
Peripheral-to-memory	AHB peripheral port	AHB memory port	DMA	possible	single	possible	possible
					burst	forbidden	
			Peripheral	forbidden	single	possible	forbidden
					burst	forbidden	
Memory-to-peripheral	AHB memory port	AHB peripheral port	DMA	possible	single	possible	possible
					burst	forbidden	
			Peripheral	forbidden	single	possible	forbidden
					burst	forbidden	
Memory-to-memory	AHB peripheral port	AHB memory port	DMA only	forbidden	single	forbidden	forbidden
					burst		

10.3.17 Stream configuration procedure

The following sequence should be followed to configure a DMA stream x (where x is the stream number):

1. If the stream is enabled, disable it by resetting the EN bit in the DMA_SxCR register, then read this bit in order to confirm that there is no ongoing stream operation. Writing this bit to 0 is not immediately effective since it is actually written to 0 once all the current transfers have finished. When the EN bit is read as 0, this means that the stream is ready to be configured. It is therefore necessary to wait for the EN bit to be cleared before starting any stream configuration. All the stream dedicated bits set in the status register (DMA_LISR and DMA_HISR) from the previous data block DMA transfer should be cleared before the stream can be re-enabled.
2. Set the peripheral port register address in the DMA_SxPAR register. The data will be moved from/ to this address to/ from the peripheral port after the peripheral event.
3. Set the memory address in the DMA_SxMA0R register (and in the DMA_SxMA1R register in the case of a double buffer mode). The data will be written to or read from this memory after the peripheral event.
4. Configure the total number of data items to be transferred in the DMA_SxNDTR register. After each peripheral event or each beat of the burst, this value is decremented.
5. Select the DMA channel (request) using CHSEL[2:0] in the DMA_SxCR register.
6. If the peripheral is intended to be the flow controller and if it supports this feature, set the PFCTRL bit in the DMA_SxCR register.
7. Configure the stream priority using the PL[1:0] bits in the DMA_SxCR register.
8. Configure the FIFO usage (enable or disable, threshold in transmission and reception)
9. Configure the data transfer direction, peripheral and memory incremented/fixed mode, single or burst transactions, peripheral and memory data widths, Circular mode,

Double buffer mode and interrupts after half and/or full transfer, and/or errors in the DMA_SxCR register.

10. Activate the stream by setting the EN bit in the DMA_SxCR register.

As soon as the stream is enabled, it can serve any DMA request from the peripheral connected to the stream.

Once half the data have been transferred on the AHB destination port, the half-transfer flag (HTIF) is set and an interrupt is generated if the half-transfer interrupt enable bit (HTIE) is set. At the end of the transfer, the transfer complete flag (TCIF) is set and an interrupt is generated if the transfer complete interrupt enable bit (TCIE) is set.

Warning: **To switch off a peripheral connected to a DMA stream request, it is mandatory to, first, switch off the DMA stream to which the peripheral is connected, then to wait for EN bit = 0. Only then can the peripheral be safely disabled.**

10.3.18 Error management

The DMA controller can detect the following errors:

- **Transfer error**: the transfer error interrupt flag (TEIFx) is set when:
  - A bus error occurs during a DMA read or a write access
  - A write access is requested by software on a memory address register in Double buffer mode whereas the stream is enabled and the current target memory is the one impacted by the write into the memory address register (refer to *Section 10.3.9: Double buffer mode*)
- **FIFO error**: the FIFO error interrupt flag (FEIFx) is set if:
  - A FIFO underrun condition is detected
  - A FIFO overrun condition is detected (no detection in memory-to-memory mode because requests and transfers are internally managed by the DMA)
  - The stream is enabled while the FIFO threshold level is not compatible with the size of the memory burst (refer to *Table 48: FIFO threshold configurations*)
- **Direct mode error**: the direct mode error interrupt flag (DMEIFx) can only be set in the peripheral-to-memory mode while operating in direct mode and when the MINC bit in the DMA_SxCR register is cleared. This flag is set when a DMA request occurs while the previous data have not yet been fully transferred into the memory (because the memory bus was not granted). In this case, the flag indicates that 2 data items were be transferred successively to the same destination address, which could be an issue if the destination is not able to manage this situation

In direct mode, the FIFO error flag can also be set under the following conditions:

- In the peripheral-to-memory mode, the FIFO can be saturated (overrun) if the memory bus is not granted for several peripheral requests
- In the memory-to-peripheral mode, an underrun condition may occur if the memory bus has not been granted before a peripheral request occurs

If the TEIFx or the FEIFx flag is set due to incompatibility between burst size and FIFO threshold level, the faulty stream is automatically disabled through a hardware clear of its EN bit in the corresponding stream configuration register (DMA_SxCR).

If the DMEIFx or the FEIFx flag is set due to an overrun or underrun condition, the faulty stream is not automatically disabled and it is up to the software to disable or not the stream by resetting the EN bit in the DMA_SxCR register. This is because there is no data loss when this kind of errors occur.

When the stream's error interrupt flag (TEIF, FEIF, DMEIF) in the DMA_LISR or DMA_HISR register is set, an interrupt is generated if the corresponding interrupt enable bit (TEIE, FEIE, DMIE) in the DMA_SxCR or DMA_SxFCR register is set.

Note: When a FIFO overrun or underrun condition occurs, the data are not lost because the peripheral request is not acknowledged by the stream until the overrun or underrun condition is cleared. If this acknowledge takes too much time, the peripheral itself may detect an overrun or underrun condition of its internal buffer and data might be lost.

10.4 DMA interrupts

For each DMA stream, an interrupt can be produced on the following events:

- Half-transfer reached
- Transfer complete
- Transfer error
- Fifo error (overrun, underrun or FIFO level error)
- Direct mode error

Separate interrupt enable control bits are available for flexibility as shown in *Table 50*.

Table 50. DMA interrupt requests

Interrupt event	Event flag	Enable control bit
Half-transfer	HTIF	HTIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE
FIFO overrun/underrun	FEIF	FEIE
Direct mode error	DMEIF	DMEIE

Note: Before setting an Enable control bit to '1', the corresponding event flag should be cleared, otherwise an interrupt is immediately generated.

10.5 DMA registers

The DMA registers have to be accessed by words (32 bits).

10.5.1 DMA low interrupt status register (DMA_LISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				TCIF3	HTIF3	TEIF3	DMEIF3	Reserved	FEIF3	TCIF2	HTIF2	TEIF2	DMEIF2	Reserved	FEIF2
r	r	r	r	r	r	r	r		r	r	r	r	r		r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				TCIF1	HTIF1	TEIF1	DMEIF1	Reserved	FEIF1	TCIF0	HTIF0	TEIF0	DMEIF0	Reserved	FEIF0
r	r	r	r	r	r	r	r		r	r	r	r	r		r

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **TCIFx**: Stream x transfer complete interrupt flag (x = 3..0)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.

0: No transfer complete event on stream x

1: A transfer complete event occurred on stream x

Bits 26, 20, 10, 4 **HTIFx**: Stream x half transfer interrupt flag (x=3..0)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.

0: No half transfer event on stream x

1: A half transfer event occurred on stream x

Bits 25, 19, 9, 3 **TEIFx**: Stream x transfer error interrupt flag (x=3..0)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.

0: No transfer error on stream x

1: A transfer error occurred on stream x

Bits 24, 18, 8, 2 **DMEIFx**: Stream x direct mode error interrupt flag (x=3..0)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.

0: No Direct Mode Error on stream x

1: A Direct Mode Error occurred on stream x

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **FEIFx**: Stream x FIFO error interrupt flag (x=3..0)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.

0: No FIFO Error event on stream x

1: A FIFO Error event occurred on stream x

10.5.2 DMA high interrupt status register (DMA_HISR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				TCIF7	HTIF7	TEIF7	DMEIF7	Reserved	FEIF7	TCIF6	HTIF6	TEIF6	DMEIF6	Reserved	FEIF6
				r	r	r	r		r	r	r	r	r		r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				TCIF5	HTIF5	TEIF5	DMEIF5	Reserved	FEIF5	TCIF4	HTIF4	TEIF4	DMEIF4	Reserved	FEIF4
				r	r	r	r		r	r	r	r	r		r

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **TCIFx**: Stream x transfer complete interrupt flag (x=7..4)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.

0: No transfer complete event on stream x

1: A transfer complete event occurred on stream x

Bits 26, 20, 10, 4 **HTIFx**: Stream x half transfer interrupt flag (x=7..4)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.

0: No half transfer event on stream x

1: A half transfer event occurred on stream x

Bits 25, 19, 9, 3 **TEIFx**: Stream x transfer error interrupt flag (x=7..4)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.

0: No transfer error on stream x

1: A transfer error occurred on stream x

Bits 24, 18, 8, 2 **DMEIFx**: Stream x direct mode error interrupt flag (x=7..4)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.

0: No Direct mode error on stream x

1: A Direct mode error occurred on stream x

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **FEIFx**: Stream x FIFO error interrupt flag (x=7..4)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.

0: No FIFO error event on stream x

1: A FIFO error event occurred on stream x

10.5.3 DMA low interrupt flag clear register (DMA_LIFCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CTCIF3	CHTIF3	CTEIF3	CDMEIF3	Reserved	CFEIF3	CTCIF2	CHTIF2	CTEIF2	CDMEIF2	Reserved	CFEIF2
				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
Reserved				CTCIF1	CHTIF1	CTEIF1	CDMEIF1	Reserved	CFEIF1	CTCIF0	CHTIF0	CTEIF0	CDMEIF0	Reserved	CFEIF0
				w	w	w	w		w	w	w	w	w		w

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **CTCIFx**: Stream x clear transfer complete interrupt flag (x = 3..0)
Writing 1 to this bit clears the corresponding TCIFx flag in the DMA_LISR register

Bits 26, 20, 10, 4 **CHTIFx**: Stream x clear half transfer interrupt flag (x = 3..0)
Writing 1 to this bit clears the corresponding HTIFx flag in the DMA_LISR register

Bits 25, 19, 9, 3 **CTEIFx**: Stream x clear transfer error interrupt flag (x = 3..0)
Writing 1 to this bit clears the corresponding TEIFx flag in the DMA_LISR register

Bits 24, 18, 8, 2 **CDMEIFx**: Stream x clear direct mode error interrupt flag (x = 3..0)
Writing 1 to this bit clears the corresponding DMEIFx flag in the DMA_LISR register

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **CFEIFx**: Stream x clear FIFO error interrupt flag (x = 3..0)
Writing 1 to this bit clears the corresponding CFEIFx flag in the DMA_LISR register

10.5.4 DMA high interrupt flag clear register (DMA_HIFCR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CTCIF7	CHTIF7	CTEIF7	CDMEIF7	Reserved	CFEIF7	CTCIF6	CHTIF6	CTEIF6	CDMEIF6	Reserved	CFEIF6
				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
Reserved				CTCIF5	CHTIF5	CTEIF5	CDMEIF5	Reserved	CFEIF5	CTCIF4	CHTIF4	CTEIF4	CDMEIF4	Reserved	CFEIF4
				w	w	w	w		w	w	w	w	w		w

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **CTCIFx**: Stream x clear transfer complete interrupt flag (x = 7..4)
Writing 1 to this bit clears the corresponding TCIFx flag in the DMA_HISR register

Bits 26, 20, 10, 4 **CHTIFx**: Stream x clear half transfer interrupt flag (x = 7..4)
Writing 1 to this bit clears the corresponding HTIFx flag in the DMA_HISR register

Bits 25, 19, 9, 3 **CTEIFx**: Stream x clear transfer error interrupt flag (x = 7..4)
Writing 1 to this bit clears the corresponding TEIFx flag in the DMA_HISR register

Bits 24, 18, 8, 2 **CDMEIFx**: Stream x clear direct mode error interrupt flag (x = 7..4)
Writing 1 to this bit clears the corresponding DMEIFx flag in the DMA_HISR register

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **CFEIFx**: Stream x clear FIFO error interrupt flag (x = 7..4)
Writing 1 to this bit clears the corresponding CFEIFx flag in the DMA_HISR register

10.5.5 DMA stream x configuration register (DMA_SxCR) (x = 0..7)

This register is used to configure the concerned stream.

Address offset: 0x10 + 0x18 × *stream number*

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CHSEL[2:0]			MBURST [1:0]		PBURST[1:0]		Reser-ved	CT	DBM	PL[1:0]	
				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
PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 27:25 **CHSEL[2:0]**: Channel selection
These bits are set and cleared by software.
000: channel 0 selected
001: channel 1 selected
010: channel 2 selected
011: channel 3 selected
100: channel 4 selected
101: channel 5 selected
110: channel 6 selected
111: channel 7 selected
These bits are protected and can be written only if EN is ‘0’

Bits 24:23 **MBURST**: Memory burst transfer configuration
These bits are set and cleared by software.
00: single transfer
01: INCR4 (incremental burst of 4 beats)
10: INCR8 (incremental burst of 8 beats)
11: INCR16 (incremental burst of 16 beats)
These bits are protected and can be written only if EN is ‘0’
In direct mode, these bits are forced to 0x0 by hardware as soon as bit EN= '1'.

Bits 22:21 **PBURST[1:0]**: Peripheral burst transfer configuration
These bits are set and cleared by software.
00: single transfer
01: INCR4 (incremental burst of 4 beats)
10: INCR8 (incremental burst of 8 beats)
11: INCR16 (incremental burst of 16 beats)
These bits are protected and can be written only if EN is ‘0’
In direct mode, these bits are forced to 0x0 by hardware.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CT**: Current target (only in double buffer mode)

This bits is set and cleared by hardware. It can also be written by software.

0: The current target memory is Memory 0 (addressed by the DMA_SxM0AR pointer)
1: The current target memory is Memory 1 (addressed by the DMA_SxM1AR pointer)
This bit can be written only if EN is ‘0’ to indicate the target memory area of the first transfer. Once the stream is enabled, this bit operates as a status flag indicating which memory area is the current target.

Bit 18 **DBM**: Double buffer mode

This bits is set and cleared by software.
0: No buffer switching at the end of transfer
1: Memory target switched at the end of the DMA transfer
This bit is protected and can be written only if EN is ‘0’.

Bits 17:16 **PL[1:0]**: Priority level

These bits are set and cleared by software.
00: Low
01: Medium
10: High
11: Very high
These bits are protected and can be written only if EN is ‘0’.

Bit 15 **PINCOS**: Peripheral increment offset size

This bit is set and cleared by software
0: The offset size for the peripheral address calculation is linked to the PSIZE
1: The offset size for the peripheral address calculation is fixed to 4 (32-bit alignment).
This bit has no meaning if bit PINC = '0'.
This bit is protected and can be written only if EN = '0'.
This bit is forced low by hardware when the stream is enabled (bit EN = '1') if the direct mode is selected or if PBURST are different from “00”.

Bits 14:13 **MSIZE[1:0]**: Memory data size

These bits are set and cleared by software.
00: byte (8-bit)
01: half-word (16-bit)
10: word (32-bit)
11: reserved
These bits are protected and can be written only if EN is ‘0’.
In direct mode, MSIZE is forced by hardware to the same value as PSIZE as soon as bit EN = '1'.

Bits 12:11 **PSIZE[1:0]**: Peripheral data size

These bits are set and cleared by software.
00: Byte (8-bit)
01: Half-word (16-bit)
10: Word (32-bit)
11: reserved
These bits are protected and can be written only if EN is ‘0’

Bit 10 **MINC**: Memory increment mode

This bit is set and cleared by software.
0: Memory address pointer is fixed
1: Memory address pointer is incremented after each data transfer (increment is done according to MSIZE)
This bit is protected and can be written only if EN is ‘0’.

Bit 9 **PINC**: Peripheral increment mode

This bit is set and cleared by software.

0: Peripheral address pointer is fixed

1: Peripheral address pointer is incremented after each data transfer (increment is done according to PSIZE)

This bit is protected and can be written only if EN is ‘0’.

Bit 8 **CIRC**: Circular mode

This bit is set and cleared by software and can be cleared by hardware.

0: Circular mode disabled

1: Circular mode enabled

When the peripheral is the flow controller (bit PFCTRL=1) and the stream is enabled (bit EN=1), then this bit is automatically forced by hardware to 0.

It is automatically forced by hardware to 1 if the DBM bit is set, as soon as the stream is enabled (bit EN ='1').

Bits 7:6 **DIR[1:0]**: Data transfer direction

These bits are set and cleared by software.

00: Peripheral-to-memory

01: Memory-to-peripheral

10: Memory-to-memory

11: reserved

These bits are protected and can be written only if EN is ‘0’.

Bit 5 **PFCTRL**: Peripheral flow controller

This bit is set and cleared by software.

0: The DMA is the flow controller

1: The peripheral is the flow controller

This bit is protected and can be written only if EN is ‘0’.

When the memory-to-memory mode is selected (bits DIR[1:0]=10), then this bit is automatically forced to 0 by hardware.

Bit 4 **TCIE**: Transfer complete interrupt enable

This bit is set and cleared by software.

0: TC interrupt disabled

1: TC interrupt enabled

Bit 3 **HTIE**: Half transfer interrupt enable

This bit is set and cleared by software.

0: HT interrupt disabled

1: HT interrupt enabled

Bit 2 **TEIE**: Transfer error interrupt enable

This bit is set and cleared by software.

0: TE interrupt disabled

1: TE interrupt enabled

Bit 1 **DMEIE**: Direct mode error interrupt enable

This bit is set and cleared by software.

0: DME interrupt disabled

1: DME interrupt enabled

Bit 0 **EN**: Stream enable / flag stream ready when read low

This bit is set and cleared by software.

0: Stream disabled

1: Stream enabled

This bit may be cleared by hardware:

- on a DMA end of transfer (stream ready to be configured)
- if a transfer error occurs on the AHB master buses
- when the FIFO threshold on memory AHB port is not compatible with the size of the burst

When this bit is read as 0, the software is allowed to program the Configuration and FIFO bits registers. It is forbidden to write these registers when the EN bit is read as 1.

Note: Before setting EN bit to '1' to start a new transfer, the event flags corresponding to the stream in DMA_LISR or DMA_HISR register must be cleared.

10.5.6 DMA stream x number of data register (DMA_SxNDTR) (x = 0..7)

Address offset: 0x14 + 0x18 × *stream number*

Reset value: 0x0000 0000

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

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

Bits 15:0 **NDT[15:0]**: Number of data items to transfer

Number of data items to be transferred (0 up to 65535). This register can be written only when the stream is disabled. When the stream is enabled, this register is read-only, indicating the remaining data items to be transmitted. This register decrements after each DMA transfer.

Once the transfer has completed, this register can either stay at zero (when the stream is in normal mode) or be reloaded automatically with the previously programmed value in the following cases:

- when the stream is configured in Circular mode.
- when the stream is enabled again by setting EN bit to '1'

If the value of this register is zero, no transaction can be served even if the stream is enabled.

10.5.7 DMA stream x peripheral address register (DMA_SxPAR) (x = 0..7)

Address offset: 0x18 + 0x18 × *stream number*

Reset value: 0x0000 0000

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

Bits 31:0 **PAR[31:0]**: Peripheral address

Base address of the peripheral data register from/to which the data will be read/written.
These bits are write-protected and can be written only when bit EN = '0' in the DMA_SxCR register.

10.5.8 DMA stream x memory 0 address register (DMA_SxM0AR) (x = 0..7)

Address offset: 0x1C + 0x18 × *stream number*

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **M0A[31:0]**: Memory 0 address

Base address of Memory area 0 from/to which the data will be read/written.
These bits are write-protected. They can be written only if:

- the stream is disabled (bit EN= '0' in the DMA_SxCR register) or
- the stream is enabled (EN='1' in DMA_SxCR register) and bit CT = '1' in the DMA_SxCR register (in Double buffer mode).

10.5.9 DMA stream x memory 1 address register (DMA_SxM1AR) (x = 0..7)

Address offset: 0x20 + 0x18 × *stream number*

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M1A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M1A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **M1A[31:0]**: Memory 1 address (used in case of Double buffer mode)

Base address of Memory area 1 from/to which the data will be read/written.

This register is used only for the Double buffer mode.

These bits are write-protected. They can be written only if:

– the stream is disabled (bit EN= '0' in the DMA_SxCR register) or

– the stream is enabled (EN='1' in DMA_SxCR register) and bit CT = '0' in the DMA_SxCR register.

10.5.10 DMA stream x FIFO control register (DMA_SxFCR) (x = 0..7)

Address offset: 0x24 + 0x24 × *stream number*

Reset value: 0x0000 0021

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

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

Bit 7 **FEIE**: FIFO error interrupt enable

This bit is set and cleared by software.

0: FE interrupt disabled

1: FE interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bits 5:3 **FS[2:0]**: FIFO status

These bits are read-only.
000: 0 < fifo_level < 1/4
001: 1/4 ≤ fifo_level < 1/2
010: 1/2 ≤ fifo_level < 3/4
011: 3/4 ≤ fifo_level < full
100: FIFO is empty
101: FIFO is full
others: no meaning
These bits are not relevant in the direct mode (DMDIS bit is zero).

Bit 2 **DMDIS**: Direct mode disable

This bit is set and cleared by software. It can be set by hardware.
0: Direct mode enabled
1: Direct mode disabled
This bit is protected and can be written only if EN is ‘0’.
This bit is set by hardware if the memory-to-memory mode is selected (DIR bit in DMA_SxCR are “10”) and the EN bit in the DMA_SxCR register is ‘1’ because the direct mode is not allowed in the memory-to-memory configuration.

Bits 1:0 **FTH[1:0]**: FIFO threshold selection

These bits are set and cleared by software.
00: 1/4 full FIFO
01: 1/2 full FIFO
10: 3/4 full FIFO
11: full FIFO
These bits are not used in the direct mode when the DMIS value is zero.
These bits are protected and can be written only if EN is ‘0’.

10.5.11 DMA register map

Table 51 summarizes the DMA registers.

Table 51. DMA register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	**DMA_LISR**	Reserved				TCIF3	HTIF3	TEIF3	DMEIF3	Reserved	FEIF3	TCIF2	HTIF2	TEIF2	DMEIF2	Reserved	FEIF2	Reserved				TCIF1	HTIF1	TEIF1	DMEIF1	Reserved	FEIF1	TCIF0	HTIF0	TEIF0	DMEIF0	Reserved	FEIF0
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x0004	**DMA_HISR**	Reserved				TCIF7	HTIF7	TEIF7	DMEIF7	Reserved	FEIF7	TCIF6	HTIF6	TEIF6	DMEIF6	Reserved	FEIF6	Reserved				TCIF5	HTIF5	TEIF5	DMEIF5	Reserved	FEIF5	TCIF4	HTIF4	TEIF4	DMEIF4	Reserved	FEIF4
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x0008	**DMA_LIFCR**	Reserved				CTCIF3	CHTIF3	TEIF3	CDMEIF3	Reserved	CFEIF3	CTCIF2	CHTIF2	CTEIF2	CDMEIF2	Reserved	CFEIF2	Reserved				CTCIF1	CHTIF1	CTEIF1	CDMEIF1	Reserved	CFEIF1	CTCIF0	CHTIF0	CTEIF0	CDMEIF0	Reserved	CFEIF0
	Reset value					0	0	0	0		0	0	0	0	0	Reserved	0					0	0	0	0	Reserved	0	0	0	0	0	Reserved	0
0x000C	**DMA_HIFCR**	Reserved				CTCIF7	CHTIF7	CTEIF7	CDMEIF7	Reserved	CFEIF7	CTCIF6	CHTIF6	CTEIF6	CDMEIF6	-	CFEIF6	Reserved				CTCIF5	CHTIF5	CTEIF5	CDMEIF5	-	CFEIF5	CTCIF4	CHTIF4	CTEIF4	CDMEIF4	-	CFEIF4
	Reset value					0	0	0	0		0	0	0	0	0	-	0					0	0	0	0	-	0	0	0	0	0	-	0
0x0010	**DMA_S0CR**	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	**DMA_S0NDTR**	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	**DMA_S0PAR**	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
0x001C	**DMA_S0M0AR**	M0A[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
0x0020	**DMA_S0M1AR**	M1A[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
0x0024	**DMA_S0FCR**	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0028	**DMA_S1CR**	Reserved				CHSEL [2:0]			MBURST[1:]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	**DMA_S1NDTR**	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51. DMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0030	**DMA_S1PAR**	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
0x0034	**DMA_S1M0A**R	M0A[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
0x0038	**DMA_S1M1AR**	M1A[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
0x003C	**DMA_S1FCR**	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0040	**DMA_S2CR**	Reserved				CHSEL [2:0]			MBURST[1:0]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR [1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	**DMA_S2NDTR**	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0048	**DMA_S2PAR**	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
0x004C	**DMA_S2M0AR**	M0A[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
0x0050	**DMA_S2M1AR**	M1A[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
0x0054	**DMA_S2FCR**	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0058	**DMA_S3CR**	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x005C	**DMA_S3NDTR**	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0060	**DMA_S3PAR**	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
0x0064	**DMA_S3M0AR**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51. DMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0068	**DMA_S3M1AR**	M1A[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
0x006C	**DMA_S3FCR**	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0070	**DMA_S4CR**	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR [1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0074	**DMA_S4NDTR**	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0078	**DMA_S4PAR**	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
0x007C	**DMA_S4M0AR**	M0A[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
0x0080	**DMA_S4M1AR**	M1A[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	**DMA_S4FCR**	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0088	**DMA_S5CR**	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008C	**DMA_S5NDTR**	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0090	**DMA_S5PAR**	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
0x0094	**DMA_S5M0AR**	M0A[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
0x0098	**DMA_S5M1AR**	M1A[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
0x009C	**DMA_S5FCR**	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x00A0	**DMA_S6CR**	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A4	**DMA_S6NDTR**	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A8	**DMA_S6PAR**	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

Table 51. DMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00AC	**DMA_S6M0AR**	M0A[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	**DMA_S6M1AR**	M1A[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
0x00B4	**DMA_S6FCR**	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x00B8	**DMA_S7CR**	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00BC	**DMA_S7NDTR**	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C0	**DMA_S7PAR**	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
0x00C4	**DMA_S7M0AR**	M0A[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
0x00C8	**DMA_S7M1AR**	M1A[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
0x00CC	**DMA_S7FCR**	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1

Refer to *Section 2.3: Memory map* for the register boundary addresses.

11 Chrom-Art Accelerator™ controller (DMA2D)

11.1 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).

11.2 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
- 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
- 4 operating modes: register-to-memory, memory-to-memory, memory-to-memory with pixel format conversion, and memory-to-memory with pixel format conversion and blending
- 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
- 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

11.3 DMA2D functional description

11.3.1 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.

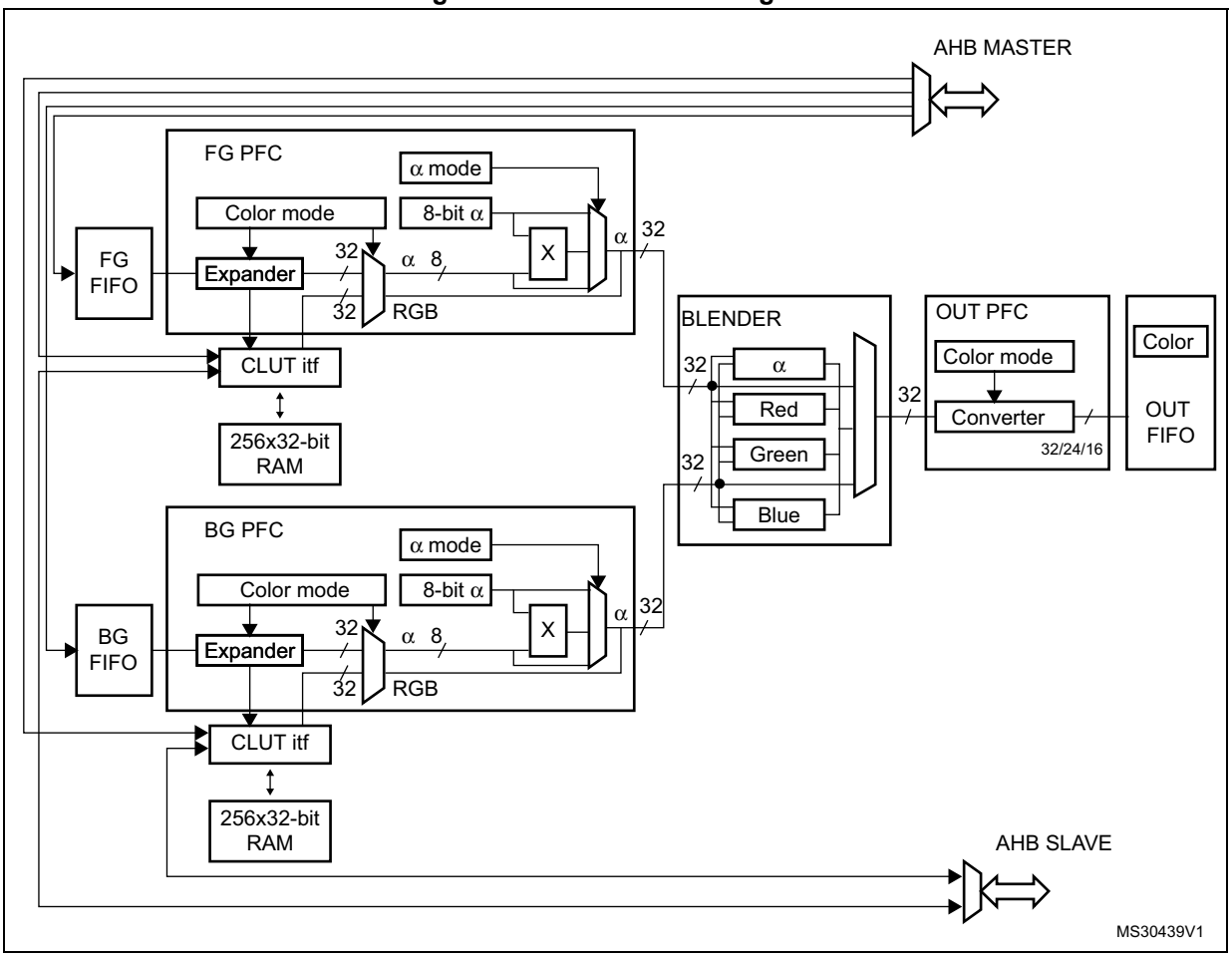
The DMA2D can operate in the following modes:

- Register-to-memory
- Memory-to-memory
- Memory-to-memory with Pixel Format Conversion
- Memory-to-memory with Pixel Format Conversion and Blending

The AHB slave port is used to program the DMA2D controller.

The block diagram of the DMA2D is shown in *Figure 40: DMA2D block diagram*.

Figure 40. DMA2D block diagram



11.3.2 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

11.3.3 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.

11.3.4 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 52: Supported color mode in input*.

Table 52. 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, 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 does not include an alpha channel, the alpha value is automatically set to 0xFF (opaque).

If the original format is indirect color mode, a CLUT is required and each pixel format converter is associated with a 256 entry 32-bit CLUT.

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 53: Data order in memory*.

Table 53. Data order in memory

Color Mode	@ + 3	@ + 2	@ + 1	@ + 0
ARGB8888	A_0[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]
RGB565	R_1[4:0]G_1[5:3]	G_1[2:0]B_1[4:0]	R_0[4:0]G_0[5:3]	G_0[2:0]B_0[4:0]
ARGB1555	A_1[0]R_1[4:0]G_1[4:3]	G_1[2:0]B_1[4:0]	A_0[0]R_0[4:0]G_0[4:3]	G_0[2:0]B_0[4:0]
ARGB4444	A_1[3:0]R_1[3:0]	G_1[3:0]B_1[3:0]	A_0[3:0]R_0[3:0]	G_0[3:0]B_0[3:0]
L8	L_3[7:0]	L_2[7:0]	L_1[7:0]	L_0[7:0]
AL44	A_3[3:0]L_3[3:0]	A_2[3:0]L_2[3:0]	A_1[3:0]L_1[3:0]	A_0[3:0]L_0[3:0]
AL88	A_1[7:0]	L_1[7:0]	A_0[7:0]	L_0[7:0]
L4	L_7[3:0]L_6[3:0]	L_5[3:0]L_4[3:0]	L_3[3:0]L_2[3:0]	L_1[3:0]L_0[3:0]
A8	A_3[7:0]	A_2[7:0]	A_1[7:0]	A_0[7:0]
A4	A_7[3:0]A_6[3:0]	A_5[3:0]A_4[3:0]	A_3[3:0]A_2[3:0]	A_1[3:0]A_0[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 54: 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 54. 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

11.3.5 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 55: Supported CLUT color mode*.

Table 55. 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 56: CLUT data order in memory*.

Table 56. CLUT data order in memory

CLUT Color Mode	@ + 3	@ + 2	@ + 1	@ + 0
ARGB8888	A_0[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]

11.3.6 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:

$$\text{with } \alpha_{Mult} = \frac{\alpha_{FG} \cdot \alpha_{BG}}{255}$$

$$\alpha_{OUT} = \alpha_{FG} + \alpha_{BG} - \alpha_{Mult}$$

$$C_{OUT} = \frac{C_{FG}.\alpha_{FG} + C_{BG}.\alpha_{BG} - C_{BG}.\alpha_{Mult}}{\alpha_{OUT}} \quad \text{with C = R or G or B}$$

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[1:0] field of the DMA2D_CR register.

11.3.7 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 57: Supported color mode in output*

Table 57. Supported color mode in output

CM[2:0]	Color mode
000	ARGB8888
001	RGB888
010	RGB565
011	ARGB1555
100	ARGB4444

11.3.8 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 58: Data order in memory*

Table 58. Data order in memory

Color Mode	@ + 3	@ + 2	@ + 1	@ + 0
ARGB8888	A_0[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]
RGB565	R_1[4:0]G_1[5:3]	G_1[2:0]B_1[4:0]	R_0[4:0]G_0[5:3]	G_0[2:0]B_0[4:0]
ARGB1555	A_1[0]R_1[4:0]G_1[4:3]	G_1[2:0]B_1[4:0]	A_0[0]R_0[4:0]G_0[4:3]	G_0[2:0]B_0[4:0]
ARGB4444	A_1[3:0]R_1[3:0]	G_1[3:0]B_1[3:0]	A_0[3:0]R_0[3:0]	G_0[3:0]B_0[3:0]

The RGB888 aligned on 32-bit is supported through the ARGB8888 mode.

11.3.9 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).

11.3.10 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.

11.3.11 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[1:0] bits of the DMA2D_CR register:

- Register-to-memory
- Memory-to-memory
- Memory-to-memory with PFC
- Memory-to-memory with PFC and blending

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:

$$\text{with } \alpha_{Mult} = \frac{\alpha_{FG} \cdot \alpha_{BG}}{255}$$

$$\alpha_{OUT} = \alpha_{FG} + \alpha_{BG} - \alpha_{Mult}$$

$$C_{OUT} = \frac{C_{FG}.\alpha_{FG} + C_{BG}.\alpha_{BG} - C_{BG}.\alpha_{Mult}}{\alpha_{OUT}} \quad \text{with C = R or G or B}$$

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.

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 not aligned with CCM of DMA2D_FGPFCCR.
- Background CLUT automatic loading: MA of DMA2D_BGCMAR not aligned with CCM of DMA2D_BGPFCCR
- Memory transfer (except in register-to-memory mode): MA of DMA2D_FGMAR not aligned with CM of DMA2D_FGPFCCR
- Memory transfer (except in register-to-memory mode): CM in DMA2D_FGPFCCR are invalid
- Memory transfer (except in register-to-memory mode): PL bits of DMA2D_NLR odd while CM of DMA2D_FGPFCCR is A4 or L4
- Memory transfer (except in register-to-memory mode): LO bits in DMA2D_FGOR odd while CM of DMA2D_FGPFCCR is A4 or L4
- Memory transfer (only in blending mode): MA bits in DMA2D_BGMAR are not aligned with the CM of DMA2D_BGPFCCR
- Memory transfer: CM of DMA2D_BGPFCCR invalid (only in blending mode)
- Memory transfer (only in blending mode): PL bits of DMA2D_NLR odd while CM of DMA2D_BGPFCCR is A4 or L4
- Memory transfer (only in blending mode): LO bits of DMA2D_BGOR odd while CM of DMA2D_BGPFCCR is A4 or L4
- Memory transfer (except in memory to memory mode): MA bits in DMA2D_OMAR are not aligned with CM bits in DMA2D_OPFCCR.
- Memory transfer (except in memory to memory mode): CM bits in DMA2D_OPFCCR invalid
- Memory transfer: NL bits in DMA2D_NLR = 0
- Memory transfer: PL bits in DMA2D_NLR = 0

11.3.12 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.

11.3.13 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.

11.3.14 Error management

Two kind of errors can be triggered:

- AHB master port errors signalled 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).

11.3.15 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.

11.4 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 59. DMA2D interrupt requests

Interrupt event	Event flag	Enable control bit
Configuration error	CEIF	CEIE
CLUT transfer complete	CTCIF	CTCIE

Table 59. DMA2D interrupt requests (continued)

Interrupt event	Event flag	Enable control bit
CLUT access error	CAEIF	CAEIE
Transfer watermark	TWF	TWIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE

11.5 DMA2D registers

11.5.1 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
Reserved														MODE	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CEIE	CTCIE	CAEIE	TWIE	TCIE	TEIE	Reserved					ABORT	SUSP	START
		rw	rw	rw	rw	rw	rw						rs	rw	rs

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

Bits 17:16 **MODE**: DMA2D mode
This bit is set and cleared by software. It cannot be modified while a transfer is ongoing.
00: Memory-to-memory (FG fetch only)
01: Memory-to-memory with PFC (FG fetch only with FG PFC active)
10: Memory-to-memory with blending (FG and BG fetch with PFC and blending)
11: Register-to-memory (no FG nor BG, only output stage active)

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

Bits 7:3 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).

11.5.2 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										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).

11.5.3 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										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

11.5.4 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.

11.5.5 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		LO[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 13:0 **LO[13: 0]**: Line offset

Line offset used for the foreground expressed in pixel. This value is used to generate the address. 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 a data transfer has started, they become read-only.

If the image format is 4-bit per pixel, the line offset must be even.

11.5.6 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.

11.5.7 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		LO[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 13:0 **LO[13: 0]**: Line offset

Line offset used for the background image (expressed in pixel). 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.

11.5.8 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]								Reserved						AM[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
CS[7:0]								Reserved		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 23:18 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

11.5.9 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
Reserved								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.

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.

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.

11.5.10 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]								Reserved						AM[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
CS[7:0]								Reserved		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 23:18 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

11.5.11 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
Reserved								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 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.

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.

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.

11.5.12 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]															
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 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.

11.5.13 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.

11.5.14 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													CM[2:0]		
													rw	rw	rw

Bits 31: 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

11.5.15 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]							
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
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]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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.

11.5.16 DMA2D output memory address register (DMA2D_OMAR)

Address offset: 0x003C

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 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.

11.5.17 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		LO[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 13:0 **LO[13: 0]**: Line Offset

Line offset used for the output (expressed in pixels). 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 the transfer has started, they are read-only.

11.5.18 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
Reserved		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.

11.5.19 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LW[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 **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.

11.5.20 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DT[7:0]								Reserved							EN
rw	rw	rw	rw	rw	rw	rw	rw								rw

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.

11.5.21 DMA2D register map

The following table summarizes the DMA2D registers. Refer to *Section 2.3: Memory map* for the DMA2D register base address.

Table 60. 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	Reserved														MODE[1:0]		Reserved			CEIE	CTCIE	CAEIE	TWIE	TCIE	TEIE	Reserved				ABORT	SUSP	START
	Reset value															0	0				0	0	0	0	0	0					0	0	0
0x0004	DMA2D_ISR	Reserved																										CEIF	CTCIF	CAEIF	TWIF	TCIF	TEIF
	Reset value																											0	0	0	0	0	0
0x0008	DMA2D_IFCR	Reserved																										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	Reserved																			LO[13:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
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	Reserved																			LO[13:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	DMA2D_FGPFCCR	ALPHA[7:0]								Reserved						AM[1:0]		CS[7:0]									Res	START	CCM	CM[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
0x0020	DMA2D_FGCOLR	APLHA[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
0x0024	DMA2D_BGPFCCR	ALPHA[7:0]								Reserved						AM[1:0]		CS[7:0]									Res	START	CCM	CM[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
0x0028	DMA2D_BGCOLR	APLHA[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
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	Reserved																													CM[2:0]		
	Reset value																														0	0	0
0x0038	DMA2D_OCOLR	APLHA[7:0]								RED[7:0]								GREEN[7:0]									BLUE[7:0]						
		Reserved																RED[4:0]			GREEN[6:0]						BLUE[4:0]						
		Reserved																A	RED[4:0]					GREEN[4:0]			BLUE[4:0]						
		Reserved																ALPHA[3:0]			RED[3:0]				GREEN[3:0]			BLUE[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
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

Table 60. 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
0x0040	DMA2D_OOR	Reserved																		LO[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	DMA2D_NLR	Res	PL[13:0]															NL[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
0x0048	DMA2D_LWR	Reserved																LW[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004C	DMA2D_AMTCR	Reserved																DT[7:0]								Reserved							EN
	Reset value																	0	0	0	0	0	0	0	0								0
0x0050-Ox03FF	-	Reserved																															
0x0400-0x07FF	DMA2D_FGCLUT	APLHA[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	APLHA[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

12 Interrupts and events

This Section applies to the whole STM32F4xx family, unless otherwise specified.

12.1 Nested vectored interrupt controller (NVIC)

12.1.1 NVIC features

The nested vector interrupt controller NVIC includes the following features:

- 82 maskable interrupt channels for STM32F405xx/07xx and STM32F415xx/17xx, and up to 91 maskable interrupt channels for STM32F42xxx and STM32F43xxx (not including the 16 interrupt lines of Cortex®-M4 with FPU)
- 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 programming manual PM0214.

12.1.2 SysTick calibration value register

The SysTick calibration value is fixed to 18750, which gives a reference time base of 1 ms with the SysTick clock set to 18.75 MHz (HCLK/8, with HCLK set to 150 MHz).

12.1.3 Interrupt and exception vectors

See *Table 62* and *Table 62*, for the vector table for the STM32F405xx/07xx and STM32F415xx/17xx and STM32F42xxx and STM32F43xxx devices.

12.2 External interrupt/event controller (EXTI)

The external interrupt/event controller consists of up to 23 edge detectors for generating event/interrupt requests. Each input line can be independently configured to select the type (interrupt or event) and the corresponding trigger event (rising or falling or both). Each line can also masked independently. A pending register maintains the status line of the interrupt requests.

Table 61. Vector table for STM32F405xx/07xx and STM32F415xx/17xx

Position	Priority	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000 0000
	-3	fixed	Reset	Reset	0x0000 0004
	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000 0008
	-1	fixed	HardFault	All class of fault	0x0000 000C
	0	settable	MemManage	Memory management	0x0000 0010
	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
	2	settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
	-	-	-	Reserved	0x0000 001C - 0x0000 002B
	3	settable	SVCall	System service call via SWI instruction	0x0000 002C
	4	settable	Debug Monitor	Debug Monitor	0x0000 0030
	-	-	-	Reserved	0x0000 0034
	5	settable	PendSV	Pendable request for system service	0x0000 0038
	6	settable	SysTick	System tick timer	0x0000 003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000 0040
1	8	settable	PVD	PVD through EXTI line detection interrupt	0x0000 0044
2	9	settable	TAMP_STAMP	Tamper and TimeStamp interrupts through the EXTI line	0x0000 0048
3	10	settable	RTC_WKUP	RTC Wakeup interrupt through the EXTI line	0x0000 004C
4	11	settable	FLASH	Flash global interrupt	0x0000 0050
5	12	settable	RCC	RCC global interrupt	0x0000 0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000 0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000 005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000 0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000 0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000 0068
11	18	settable	DMA1_Stream0	DMA1 Stream0 global interrupt	0x0000 006C

Table 61. Vector table for STM32F405xx/07xx and STM32F415xx/17xx (continued)

Position	Priority	Type of priority	Acronym	Description	Address
12	19	settable	DMA1_Stream1	DMA1 Stream1 global interrupt	0x0000 0070
13	20	settable	DMA1_Stream2	DMA1 Stream2 global interrupt	0x0000 0074
14	21	settable	DMA1_Stream3	DMA1 Stream3 global interrupt	0x0000 0078
15	22	settable	DMA1_Stream4	DMA1 Stream4 global interrupt	0x0000 007C
16	23	settable	DMA1_Stream5	DMA1 Stream5 global interrupt	0x0000 0080
17	24	settable	DMA1_Stream6	DMA1 Stream6 global interrupt	0x0000 0084
18	25	settable	ADC	ADC1, ADC2 and ADC3 global interrupts	0x0000 0088
19	26	settable	CAN1_TX	CAN1 TX interrupts	0x0000 008C
20	27	settable	CAN1_RX0	CAN1 RX0 interrupts	0x0000 0090
21	28	settable	CAN1_RX1	CAN1 RX1 interrupt	0x0000 0094
22	29	settable	CAN1_SCE	CAN1 SCE interrupt	0x0000 0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000 009C
24	31	settable	TIM1_BRK_TIM9	TIM1 Break interrupt and TIM9 global interrupt	0x0000 00A0
25	32	settable	TIM1_UP_TIM10	TIM1 Update interrupt and TIM10 global interrupt	0x0000 00A4
26	33	settable	TIM1_TRG_COM_TIM11	TIM1 Trigger and Commutation interrupts and TIM11 global interrupt	0x0000 00A8
27	34	settable	TIM1_CC	TIM1 Capture Compare interrupt	0x0000 00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000 00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000 00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000 00B8
31	38	settable	I2C1_EV	I^2C1 event interrupt	0x0000 00BC
32	39	settable	I2C1_ER	I^2C1 error interrupt	0x0000 00C0
33	40	settable	I2C2_EV	I^2C2 event interrupt	0x0000 00C4
34	41	settable	I2C2_ER	I^2C2 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

Table 61. Vector table for STM32F405xx/07xx and STM32F415xx/17xx (continued)

Position	Priority	Type of priority	Acronym	Description	Address
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000 00E0
41	48	settable	RTC_Alarm	RTC Alarms (A and B) through EXTI line interrupt	0x0000 00E4
42	49	settable	OTG_FS_WKUP	USB On-The-Go FS Wakeup through EXTI line interrupt	0x0000 00E8
43	50	settable	TIM8_BRK_TIM12	TIM8 Break interrupt and TIM12 global interrupt	0x0000 00EC
44	51	settable	TIM8_UP_TIM13	TIM8 Update interrupt and TIM13 global interrupt	0x0000 00F0
45	52	settable	TIM8_TRG_COM_TIM14	TIM8 Trigger and Commutation interrupts and TIM14 global interrupt	0x0000 00F4
46	53	settable	TIM8_CC	TIM8 Capture Compare interrupt	0x0000 00F8
47	54	settable	DMA1_Stream7	DMA1 Stream7 global interrupt	0x0000 00FC
48	55	settable	FSMC	FSMC global interrupt	0x0000 0100
49	56	settable	SDIO	SDIO global interrupt	0x0000 0104
50	57	settable	TIM5	TIM5 global interrupt	0x0000 0108
51	58	settable	SPI3	SPI3 global interrupt	0x0000 010C
52	59	settable	UART4	UART4 global interrupt	0x0000 0110
53	60	settable	UART5	UART5 global interrupt	0x0000 0114
54	61	settable	TIM6_DAC	TIM6 global interrupt, DAC1 and DAC2 underrun error interrupts	0x0000 0118
55	62	settable	TIM7	TIM7 global interrupt	0x0000 011C
56	63	settable	DMA2_Stream0	DMA2 Stream0 global interrupt	0x0000 0120
57	64	settable	DMA2_Stream1	DMA2 Stream1 global interrupt	0x0000 0124
58	65	settable	DMA2_Stream2	DMA2 Stream2 global interrupt	0x0000 0128
59	66	settable	DMA2_Stream3	DMA2 Stream3 global interrupt	0x0000 012C
60	67	settable	DMA2_Stream4	DMA2 Stream4 global interrupt	0x0000 0130
61	68	settable	ETH	Ethernet global interrupt	0x0000 0134
62	69	settable	ETH_WKUP	Ethernet Wakeup through EXTI line interrupt	0x0000 0138
63	70	settable	CAN2_TX	CAN2 TX interrupts	0x0000 013C
64	71	settable	CAN2_RX0	CAN2 RX0 interrupts	0x0000 0140

Table 61. Vector table for STM32F405xx/07xx and STM32F415xx/17xx (continued)

Position	Priority	Type of priority	Acronym	Description	Address
65	72	settable	CAN2_RX1	CAN2 RX1 interrupt	0x0000 0144
66	73	settable	CAN2_SCE	CAN2 SCE interrupt	0x0000 0148
67	74	settable	OTG_FS	USB On The Go FS global interrupt	0x0000 014C
68	75	settable	DMA2_Stream5	DMA2 Stream5 global interrupt	0x0000 0150
69	76	settable	DMA2_Stream6	DMA2 Stream6 global interrupt	0x0000 0154
70	77	settable	DMA2_Stream7	DMA2 Stream7 global interrupt	0x0000 0158
71	78	settable	USART6	USART6 global interrupt	0x0000 015C
72	79	settable	I2C3_EV	I^2C3 event interrupt	0x0000 0160
73	80	settable	I2C3_ER	I^2C3 error interrupt	0x0000 0164
74	81	settable	OTG_HS_EP1_OUT	USB On The Go HS End Point 1 Out global interrupt	0x0000 0168
75	82	settable	OTG_HS_EP1_IN	USB On The Go HS End Point 1 In global interrupt	0x0000 016C
76	83	settable	OTG_HS_WKUP	USB On The Go HS Wakeup through EXTI interrupt	0x0000 0170
77	84	settable	OTG_HS	USB On The Go HS global interrupt	0x0000 0174
78	85	settable	DCMI	DCMI global interrupt	0x0000 0178
79	86	settable	CRYP	CRYP crypto global interrupt	0x0000 017C
80	87	settable	HASH_RNG	Hash and Rng global interrupt	0x0000 0180
81	88	settable	FPU	FPU global interrupt	0x0000 0184

Table 62. Vector table for STM32F42xxx and STM32F43xxx

Position	Priority	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000 0000
	-3	fixed	Reset	Reset	0x0000 0004
	-2	fixed	NMI	Non maskable interrupt, Clock Security System	0x0000 0008

Table 62. Vector table for STM32F42xxx and STM32F43xxx (continued)

Position	Priority	Type of priority	Acronym	Description	Address
	-1	fixed	HardFault	All class of fault	0x0000 000C
	0	settable	MemManage	Memory management	0x0000 0010
	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
	2	settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
	-	-	-	Reserved	0x0000 001C - 0x0000 002B
	3	settable	SVCall	System Service call via SWI instruction	0x0000 002C
	4	settable	Debug Monitor	Debug Monitor	0x0000 0030
		-	-	Reserved	0x0000 0034
	5	settable	PendSV	Pendable request for system service	0x0000 0038
	6	settable	Systick	System tick timer	0x0000 003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000 0040
1	8	settable	PVD	PVD through EXTI line detection interrupt	0x0000 0044
2	9	settable	TAMP_STAMP	Tamper and TimeStamp interrupts through the EXTI line	0x0000 0048
3	10	settable	RTC_WKUP	RTC Wakeup interrupt through the EXTI line	0x0000 004C
4	11	settable	FLASH	Flash global interrupt	0x0000 0050
5	12	settable	RCC	RCC global interrupt	0x0000 0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000 0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000 005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000 0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000 0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000 0068
11	18	settable	DMA1_Stream0	DMA1 Stream0 global interrupt	0x0000 006C
12	19	settable	DMA1_Stream1	DMA1 Stream1 global interrupt	0x0000 0070
13	20	settable	DMA1_Stream2	DMA1 Stream2 global interrupt	0x0000 0074
14	21	settable	DMA1_Stream3	DMA1 Stream3 global interrupt	0x0000 0078
15	22	settable	DMA1_Stream4	DMA1 Stream4 global interrupt	0x0000 007C
16	23	settable	DMA1_Stream5	DMA1 Stream5 global interrupt	0x0000 0080
17	24	settable	DMA1_Stream6	DMA1 Stream6 global interrupt	0x0000 0084

Table 62. Vector table for STM32F42xxx and STM32F43xxx (continued)

Position	Priority	Type of priority	Acronym	Description	Address
18	25	settable	ADC	ADC1, ADC2 and ADC3 global interrupts	0x0000 0088
19	26	settable	CAN1_TX	CAN1 TX interrupts	0x0000 008C
20	27	settable	CAN1_RX0	CAN1 RX0 interrupts	0x0000 0090
21	28	settable	CAN1_RX1	CAN1 RX1 interrupt	0x0000 0094
22	29	settable	CAN1_SCE	CAN1 SCE interrupt	0x0000 0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000 009C
24	31	settable	TIM1_BRK_TIM9	TIM1 Break interrupt and TIM9 global interrupt	0x0000 00A0
25	32	settable	TIM1_UP_TIM10	TIM1 Update interrupt and TIM10 global interrupt	0x0000 00A4
26	33	settable	TIM1_TRG_COM_TIM11	TIM1 Trigger and Commutation interrupts and TIM11 global interrupt	0x0000 00A8
27	34	settable	TIM1_CC	TIM1 Capture Compare interrupt	0x0000 00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000 00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000 00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000 00B8
31	38	settable	I2C1_EV	I^2C1 event interrupt	0x0000 00BC
32	39	settable	I2C1_ER	I^2C1 error interrupt	0x0000 00C0
33	40	settable	I2C2_EV	I^2C2 event interrupt	0x0000 00C4
34	41	settable	I2C2_ER	I^2C2 error interrupt	0x0000 00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000 00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000 00D0
37	44	settable	USART1	USART1 global interrupt	0x0000 00D4
38	45	settable	USART2	USART2 global interrupt	0x0000 00D8
39	46	settable	USART3	USART3 global interrupt	0x0000 00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000 00E0
41	48	settable	RTC_Alarm	RTC Alarms (A and B) through EXTI line interrupt	0x0000 00E4
42	49	settable	OTG_FS_WKUP	USB On-The-Go FS Wakeup through EXTI line interrupt	0x0000 00E8
43	50	settable	TIM8_BRK_TIM12	TIM8 Break interrupt and TIM12 global interrupt	0x0000 00EC

Table 62. Vector table for STM32F42xxx and STM32F43xxx (continued)

Position	Priority	Type of priority	Acronym	Description	Address
44	51	settable	TIM8_UP_TIM13	TIM8 Update interrupt and TIM13 global interrupt	0x0000 00F0
45	52	settable	TIM8_TRG_COM_TIM14	TIM8 Trigger and Commutation interrupts and TIM14 global interrupt	0x0000 00F4
46	53	settable	TIM8_CC	TIM8 Capture Compare interrupt	0x0000 00F8
47	54	settable	DMA1_Stream7	DMA1 Stream7 global interrupt	0x0000 00FC
48	55	settable	FSMC	FSMC global interrupt	0x0000 0100
49	56	settable	SDIO	SDIO global interrupt	0x0000 0104
50	57	settable	TIM5	TIM5 global interrupt	0x0000 0108
51	58	settable	SPI3	SPI3 global interrupt	0x0000 010C
52	59	settable	UART4	UART4 global interrupt	0x0000 0110
53	60	settable	UART5	UART5 global interrupt	0x0000 0114
54	61	settable	TIM6_DAC	TIM6 global interrupt, DAC1 and DAC2 underrun error interrupts	0x0000 0118
55	62	settable	TIM7	TIM7 global interrupt	0x0000 011C
56	63	settable	DMA2_Stream0	DMA2 Stream0 global interrupt	0x0000 0120
57	64	settable	DMA2_Stream1	DMA2 Stream1 global interrupt	0x0000 0124
58	65	settable	DMA2_Stream2	DMA2 Stream2 global interrupt	0x0000 0128
59	66	settable	DMA2_Stream3	DMA2 Stream3 global interrupt	0x0000 012C
60	67	settable	DMA2_Stream4	DMA2 Stream4 global interrupt	0x0000 0130
61	68	settable	ETH	Ethernet global interrupt	0x0000 0134
62	69	settable	ETH_WKUP	Ethernet Wakeup through EXTI line interrupt	0x0000 0138
63	70	settable	CAN2_TX	CAN2 TX interrupts	0x0000 013C
64	71	settable	CAN2_RX0	CAN2 RX0 interrupts	0x0000 0140
65	72	settable	CAN2_RX1	CAN2 RX1 interrupt	0x0000 0144
66	73	settable	CAN2_SCE	CAN2 SCE interrupt	0x0000 0148
67	74	settable	OTG_FS	USB On The Go FS global interrupt	0x0000 014C
68	75	settable	DMA2_Stream5	DMA2 Stream5 global interrupt	0x0000 0150
69	76	settable	DMA2_Stream6	DMA2 Stream6 global interrupt	0x0000 0154
70	77	settable	DMA2_Stream7	DMA2 Stream7 global interrupt	0x0000 0158
71	78	settable	USART6	USART6 global interrupt	0x0000 015C

Table 62. Vector table for STM32F42xxx and STM32F43xxx (continued)

Position	Priority	Type of priority	Acronym	Description	Address
72	79	settable	I2C3_EV	I^2C3 event interrupt	0x0000 0160
73	80	settable	I2C3_ER	I^2C3 error interrupt	0x0000 0164
74	81	settable	OTG_HS_EP1_OUT	USB On The Go HS End Point 1 Out global interrupt	0x0000 0168
75	82	settable	OTG_HS_EP1_IN	USB On The Go HS End Point 1 In global interrupt	0x0000 016C
76	83	settable	OTG_HS_WKUP	USB On The Go HS Wakeup through EXTI interrupt	0x0000 0170
77	84	settable	OTG_HS	USB On The Go HS global interrupt	0x0000 0174
78	85	settable	DCMI	DCMI global interrupt	0x0000 0178
79	86	settable	CRYP	CRYP crypto global interrupt	0x0000 017C
80	87	settable	HASH_RNG	Hash and Rng global interrupt	0x0000 0180
81	88	Settable	FPU	FPU global interrupt	0x0000 0184
82	89	settable	UART7	UART 7 global interrupt	0x0000 0188
83	90	settable	UART8	UART 8 global interrupt	0x0000 018C
84	91	settable	SPI4	SPI 4 global interrupt	0x0000 0190
85	92	settable	SPI5	SPI 5 global interrupt	0x0000 0194
86	93	settable	SPI6	SPI 6 global interrupt	0x0000 0198
87	94	settable	SAI1	SAI1 global interrupt	0x0000 019C
88	95	settable	LCD-TFT	LTDC global interrupt	0x0000 01A0
89	96	settable	LCD-TFT	LTDC global Error interrupt	0x0000 01A4
90	97	settable	DMA2D	DMA2D global interrupt	0x0000 01A8

12.2.1 EXTI main features

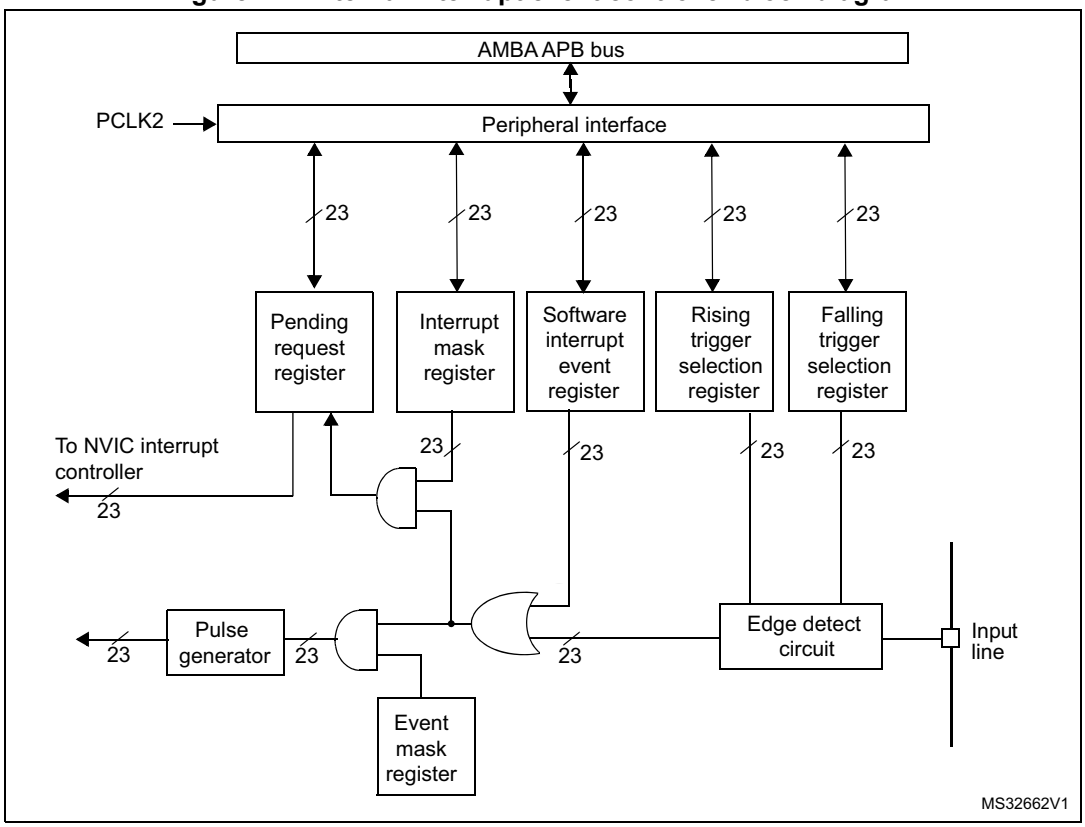
The main features of the EXTI controller are the following:

- independent trigger and mask on each interrupt/event line
- dedicated status bit for each interrupt line
- generation of up to 23 software event/interrupt requests
- detection of external signals with a pulse width lower than the APB2 clock period. Refer to the electrical characteristics section of the STM32F4xx datasheets for details on this parameter.

12.2.2 EXTI block diagram

Figure 41 shows the block diagram.

Figure 41. External interrupt/event controller block diagram



12.2.3 Wakeup event management

The STM32F4xx are able to handle external or internal events in order to wake up the core (WFE). The wakeup event can be generated either by:

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

To use an external line as a wakeup event, refer to *Section 12.2.4: Functional description*.

12.2.4 Functional description

To generate the interrupt, the interrupt line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the interrupt request by writing a ‘1’ to the corresponding bit in the interrupt mask register. When the selected edge occurs on the external interrupt line, an interrupt request is

generated. The pending bit corresponding to the interrupt line is also set. This request is reset by writing a ‘1’ in the pending register.

To generate the event, the event line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the event request by writing a ‘1’ to the corresponding bit in the event mask register. When the selected edge occurs on the event line, an event pulse is generated. The pending bit corresponding to the event line is not set.

An interrupt/event request can also be generated by software by writing a ‘1’ in the software interrupt/event register.

Hardware interrupt selection

To configure the 23 lines as interrupt sources, use the following procedure:

- Configure the mask bits of the 23 interrupt lines (EXTI_IMR)
- Configure the Trigger selection bits of the interrupt lines (EXTI_RTSR and EXTI_FTSR)
- Configure the enable and mask bits that control the NVIC IRQ channel mapped to the external interrupt controller (EXTI) so that an interrupt coming from one of the 23 lines can be correctly acknowledged.

Hardware event selection

To configure the 23 lines as event sources, use the following procedure:

- Configure the mask bits of the 23 event lines (EXTI_EMR)
- Configure the Trigger selection bits of the event lines (EXTI_RTSR and EXTI_FTSR)

Software interrupt/event selection

The 23 lines can be configured as software interrupt/event lines. The following is the procedure to generate a software interrupt.

- Configure the mask bits of the 23 interrupt/event lines (EXTI_IMR, EXTI_EMR)
- Set the required bit in the software interrupt register (EXTI_SWIER)

12.2.5 External interrupt/event line mapping

Up to 140 GPIOs (STM32F405xx/07xx and STM32F415xx/17xx), 168 GPIOs (STM32F42xxx and STM32F43xxx) are connected to the 16 external interrupt/event lines in the following manner:

Figure 42. External interrupt/event GPIO mapping (STM32F405xx/07xx and STM32F415xx/17xx)

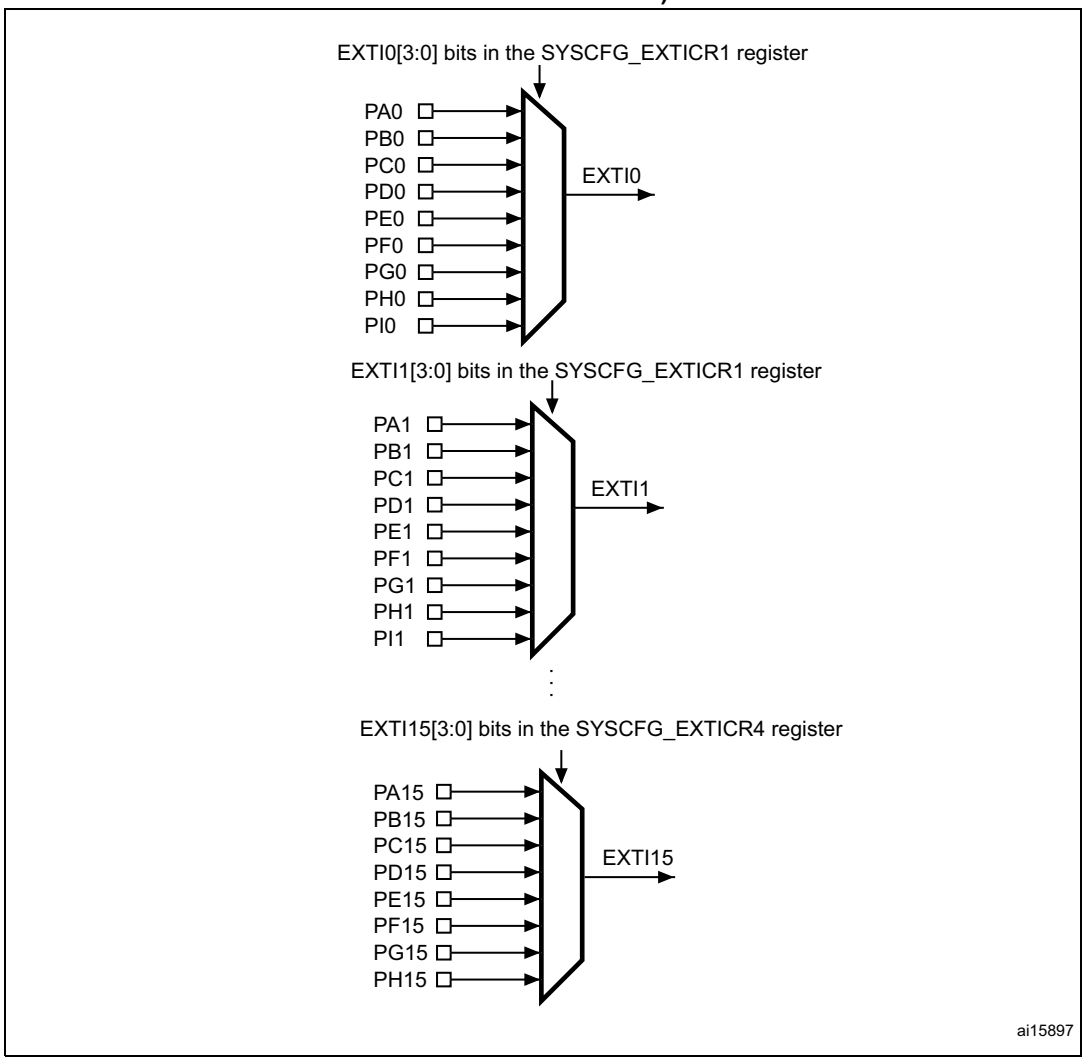
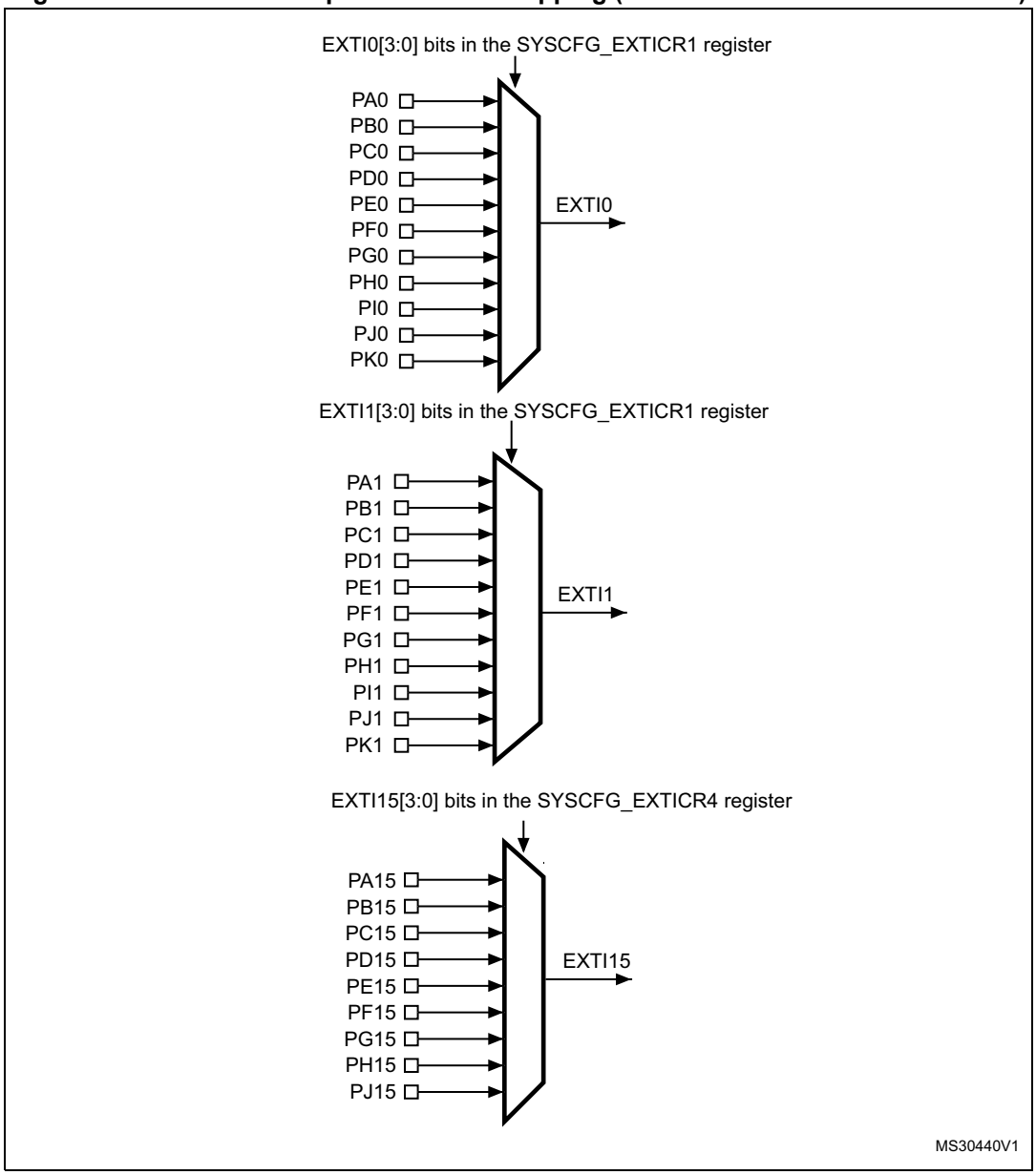


Figure 43. External interrupt/event GPIO mapping (STM32F42xxx and STM32F43xxx)



The seven other EXTI lines are connected as follows:

- EXTI line 16 is connected to the PVD output
- EXTI line 17 is connected to the RTC Alarm event
- EXTI line 18 is connected to the USB OTG FS Wakeup event
- EXTI line 19 is connected to the Ethernet Wakeup event
- EXTI line 20 is connected to the USB OTG HS (configured in FS) Wakeup event
- EXTI line 21 is connected to the RTC Tamper and TimeStamp events
- EXTI line 22 is connected to the RTC Wakeup event

12.3 EXTI registers

Refer to *Section 1.1: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

12.3.1 Interrupt mask register (EXTI_IMR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									MR22	MR21	MR20	MR19	MR18	MR17	MR16
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 22:0 **MRx:** Interrupt mask on line x

0: Interrupt request from line x is masked
1: Interrupt request from line x is not masked

12.3.2 Event mask register (EXTI_EMR)

Address offset: 0x04
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									MR22	MR21	MR20	MR19	MR18	MR17	MR16
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 22:0 **MRx:** Event mask on line x

0: Event request from line x is masked
1: Event request from line x is not masked

12.3.3 Rising trigger selection register (EXTI_RTSR)

Address offset: 0x08
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									TR22	TR21	TR20	TR19	TR18	TR17	TR16
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 22:0 **TRx:** Rising trigger event configuration bit of line x
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

Note: The external wakeup lines are edge triggered, no glitch must be generated on these lines. If a rising edge occurs on the external interrupt line while writing to the EXTI_RTSR register, the pending bit is be set.

Rising and falling edge triggers can be set for the same interrupt line. In this configuration, both generate a trigger condition.

12.3.4 Falling trigger selection register (EXTI_FTSR)

Address offset: 0x0C
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									TR22	TR21	TR20	TR19	TR18	TR17	TR16
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 22:0 **TRx:** Falling trigger event configuration bit of line x
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line.

Note: The external wakeup lines are edge triggered, no glitch must be generated on these lines. If a falling edge occurs on the external interrupt line while writing to the EXTI_FTSR register, the pending bit is not set.

Rising and falling edge triggers can be set for the same interrupt line. In this configuration, both generate a trigger condition.

12.3.5 Software interrupt event register (EXTI_SWIER)

Address offset: 0x10
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									SWIER 22	SWIER 21	SWIER 20	SWIER 19	SWIER 18	SWIER 17	SWIER 16
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIER 15	SWIER 14	SWIER 13	SWIER 12	SWIER 11	SWIER 10	SWIER 9	SWIER 8	SWIER 7	SWIER 6	SWIER 5	SWIER 4	SWIER 3	SWIER 2	SWIER 1	SWIER 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 22:0 **SWIERx:** Software Interrupt on line x

If interrupt are enabled on line x in the EXTI_IMR register, writing '1' to SWIERx bit when it is set at '0' sets the corresponding pending bit in the EXTI_PR register, thus resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit in EXTI_PR (by writing a 1 to the bit).

12.3.6 Pending register (EXTI_PR)

Address offset: 0x14
Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									PR22	PR21	PR20	PR19	PR18	PR17	PR16
									rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bits 22:0 **PRx:** Pending bit

0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line.
This bit is cleared by programming it to ‘1’.

12.3.7 EXTI register map

Table 64 gives the EXTI register map and the reset values.

Table 63. External interrupt/event controller register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**EXTI_IMR**	Reserved									MR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**EXTI_EMR**	Reserved									MR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**EXTI_RTSR**	Reserved									TR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**EXTI_FTSR**	Reserved									TR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**EXTI_SWIER**	Reserved									SWIER[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**EXTI_PR**	Reserved									PR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory map* for the register boundary addresses.

13 Analog-to-digital converter (ADC)

This section applies to the whole STM32F4xx family, unless otherwise specified.

13.1 ADC introduction

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 19 multiplexed channels allowing it to measure signals from 16 external sources, two internal sources, and the V_{BAT} channel. The A/D conversion of the channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored into a left- or right-aligned 16-bit data register.

The analog watchdog feature allows the application to detect if the input voltage goes beyond the user-defined, higher or lower thresholds.

13.2 ADC main features

- 12-bit, 10-bit, 8-bit or 6-bit configurable resolution
- Interrupt generation at the end of conversion, end of injected conversion, and in case of analog watchdog or overrun events
- Single and continuous conversion modes
- Scan mode for automatic conversion of channel 0 to channel ‘n’
- Data alignment with in-built data coherency
- Channel-wise programmable sampling time
- External trigger option with configurable polarity for both regular and injected conversions
- Discontinuous mode
- Dual/Triple mode (on devices with 2 ADCs or more)
- Configurable DMA data storage in Dual/Triple ADC mode
- Configurable delay between conversions in Dual/Triple interleaved mode
- ADC conversion type (refer to the datasheets)
- ADC supply requirements: 2.4 V to 3.6 V at full speed and down to 1.8 V at slower speed
- ADC input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$
- DMA request generation during regular channel conversion

Figure 44 shows the block diagram of the ADC.

Note: V_{REF-}, if available (depending on package), must be tied to V_{SSA}.

13.3 ADC functional description

Figure 44 shows a single ADC block diagram and *Table 65* gives the ADC pin description.

Figure 44. Single ADC block diagram

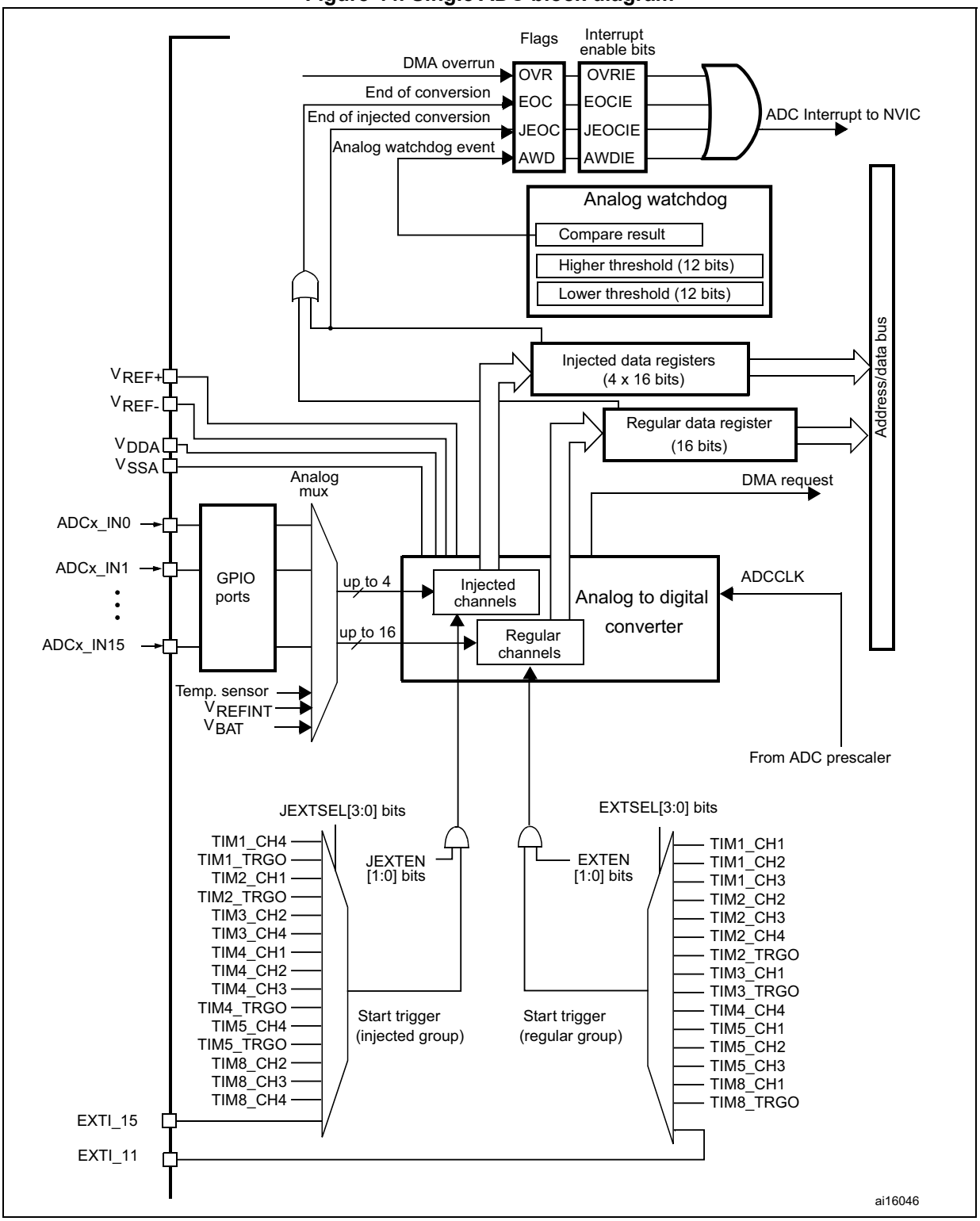


Table 65. ADC pins

Name	Signal type	Remarks
V_{REF+}	Input, analog reference positive	The higher/positive reference voltage for the ADC, 1.8 V ≤V_{REF+} ≤V_{DDA}
V_{DDA}	Input, analog supply	Analog power supply equal to V_{DD} and 2.4 V ≤V_{DDA} ≤V_{DD} (3.6 V) for full speed 1.8 V ≤V_{DDA} ≤V_{DD} (3.6 V) for reduced speed
V_{REF-}	Input, analog reference negative	The lower/negative reference voltage for the ADC, V_{REF-} = V_{SSA}
V_{SSA}	Input, analog supply ground	Ground for analog power supply equal to V_{SS}
ADCx_IN[15:0]	Analog input signals	16 analog input channels

13.3.1 ADC on-off control

The ADC is powered on by setting the ADON bit in the ADC_CR2 register. When the ADON bit is set for the first time, it wakes up the ADC from the Power-down mode.

Conversion starts when either the SWSTART or the JSWSTART bit is set.

You can stop conversion and put the ADC in power down mode by clearing the ADON bit. In this mode the ADC consumes almost no power (only a few µA).

13.3.2 ADC clock

The ADC features two clock schemes:

- Clock for the analog circuitry: ADCCLK, common to all ADCs

  This clock is generated from the APB2 clock divided by a programmable prescaler that allows the ADC to work at f_{PCLK2}/2, /4, /6 or /8. Refer to the datasheets for the maximum value of ADCCLK.
- Clock for the digital interface (used for registers read/write access)

  This clock is equal to the APB2 clock. The digital interface clock can be enabled/disabled individually for each ADC through the RCC APB2 peripheral clock enable register (RCC_APB2ENR).

13.3.3 Channel selection

There are 16 multiplexed channels. It is possible to organize the conversions in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADC_IN3, ADC_IN8, ADC_IN2, ADC_IN2, ADC_IN0, ADC_IN2, ADC_IN2, ADC_IN15.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRx registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.
- An **injected group** is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register.

The total number of conversions in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

If the ADC_SQRx or ADC_JSQR registers are modified during a conversion, the current conversion is reset and a new start pulse is sent to the ADC to convert the newly chosen group.

Temperature sensor, V_{REFINT} and V_{BAT} internal channels

- For the STM32F40x and STM32F41x devices, the temperature sensor is internally connected to channel ADC1_IN16.

  The internal reference voltage VREFINT is connected to ADC1_IN17.
- For the STM32F42x and STM32F43x devices, the temperature sensor is internally connected to ADC1_IN18 channel which is shared with VBAT. Only one conversion, temperature sensor or VBAT, must be selected at a time. When the temperature sensor and VBAT conversion are set simultaneously, only the VBAT conversion is performed.

  The internal reference voltage VREFINT is connected to ADC1_IN17.

The V_{BAT} channel (connected to channel ADC1_IN18) can also be converted as an injected or regular channel.

Note: The temperature sensor, V_{REFINT} and the V_{BAT} channel are available only on the master ADC1 peripheral.

13.3.4 Single conversion mode

In Single conversion mode the ADC does one conversion. This mode is started with the CONT bit at 0 by either:

- setting the SWSTART bit in the ADC_CR2 register (for a regular channel only)
- setting the JSWSTART bit (for an injected channel)
- external trigger (for a regular or injected channel)

Once the conversion of the selected channel is complete:

- If a regular channel was converted:
  - The converted data are stored into the 16-bit ADC_DR register
  - The EOC (end of conversion) flag is set
  - An interrupt is generated if the EOCIE bit is set
- If an injected channel was converted:
  - The converted data are stored into the 16-bit ADC_JDR1 register
  - The JEOC (end of conversion injected) flag is set
  - An interrupt is generated if the JEOCIE bit is set

Then the ADC stops.

13.3.5 Continuous conversion mode

In continuous conversion mode, the ADC starts a new conversion as soon as it finishes one. This mode is started with the CONT bit at 1 either by external trigger or by setting the SWSTRT bit in the ADC_CR2 register (for regular channels only).

After each conversion:

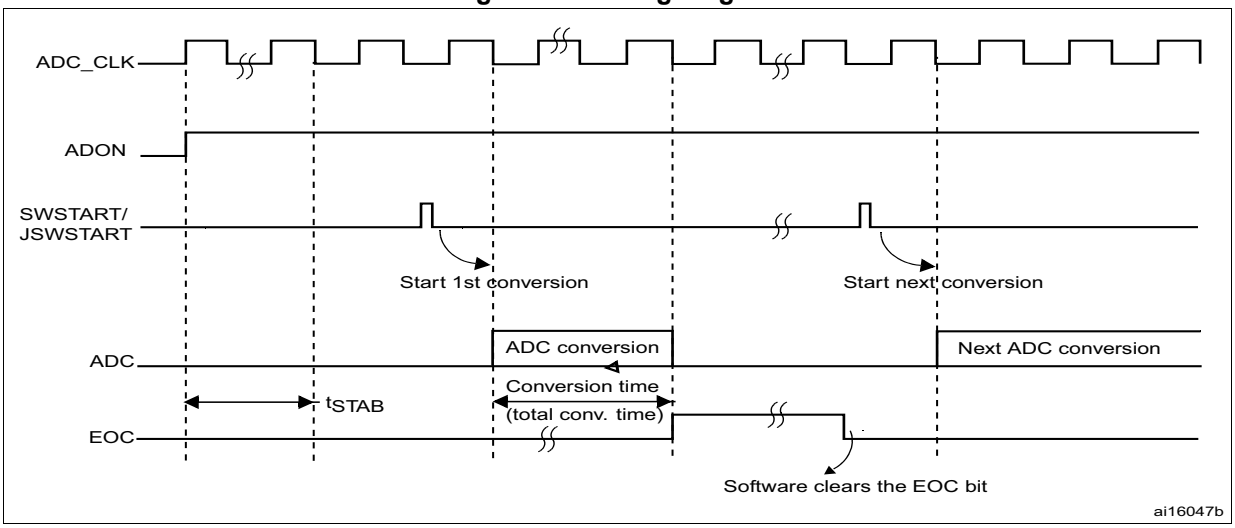
- If a regular group of channels was converted:
  - The last converted data are stored into the 16-bit ADC_DR register
  - The EOC (end of conversion) flag is set
  - An interrupt is generated if the EOCIE bit is set

Note: Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit), refer to Auto-injection section).

13.3.6 Timing diagram

As shown in *Figure 45*, the ADC needs a stabilization time of t_{STAB} before it starts converting accurately. After the start of the ADC conversion and after 15 clock cycles, the EOC flag is set and the 16-bit ADC data register contains the result of the conversion.

Figure 45. Timing diagram



13.3.7 Analog watchdog

The AWD analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold. These thresholds are programmed in the 12 least significant bits of the ADC_HTR and ADC_LTR 16-bit registers. An interrupt can be enabled by using the AWDIE bit in the ADC_CR1 register.

The threshold value is independent of the alignment selected by the ALIGN bit in the ADC_CR2 register. The analog voltage is compared to the lower and higher thresholds before alignment.

Table 66 shows how the ADC_CR1 register should be configured to enable the analog watchdog on one or more channels.

Figure 46. Analog watchdog's guarded area

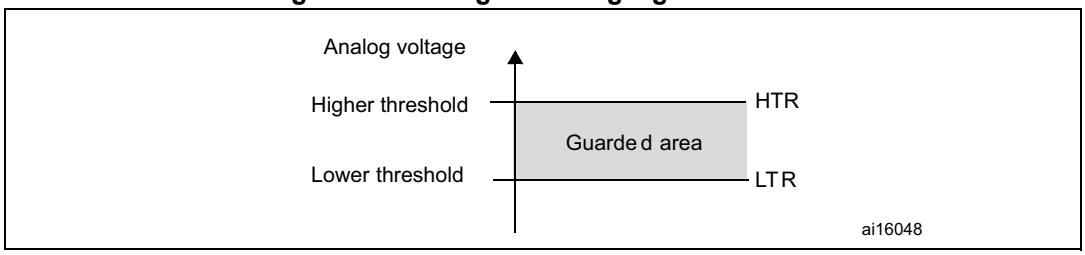


Table 66. Analog watchdog channel selection

Channels guarded by the analog watchdog	ADC_CR1 register control bits (x = don't care)		
	AWDSGL bit	AWDEN bit	JAWDEN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single(1) injected channel	1	0	1
Single(1) regular channel	1	1	0
Single (1) regular or injected channel	1	1	1

1. Selected by the AWDCH[4:0] bits

13.3.8 Scan mode

This mode is used to scan a group of analog channels.

The Scan mode is selected by setting the SCAN bit in the ADC_CR1 register. Once this bit has been set, the ADC scans all the channels selected in the ADC_SQRx registers (for regular channels) or in the ADC_JSQR register (for injected channels). A single conversion is performed for each channel of the group. After each end of conversion, the next channel in the group is converted automatically. If the CONT bit is set, regular channel conversion does not stop at the last selected channel in the group but continues again from the first selected channel.

If the DMA bit is set, the direct memory access (DMA) controller is used to transfer the data converted from the regular group of channels (stored in the ADC_DR register) to SRAM after each regular channel conversion.

The EOC bit is set in the ADC_SR register:

- At the end of each regular group sequence if the EOCS bit is cleared to 0
- At the end of each regular channel conversion if the EOCS bit is set to 1

The data converted from an injected channel are always stored into the ADC_JDRx registers.

13.3.9 Injected channel management

Triggered injection

To use triggered injection, the JAUTO bit must be cleared in the ADC_CR1 register.

1. Start the conversion of a group of regular channels either by external trigger or by setting the SWSTART bit in the ADC_CR2 register.
2. If an external injected trigger occurs or if the JSWSTART bit is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches to Scan-once mode.
3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
   If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence.
   Figure 47 shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence. For instance, if the sequence length is 30 ADC clock cycles (that is two conversions with a sampling time of 3 clock periods), the minimum interval between triggers must be 31 ADC clock cycles.

Auto-injection

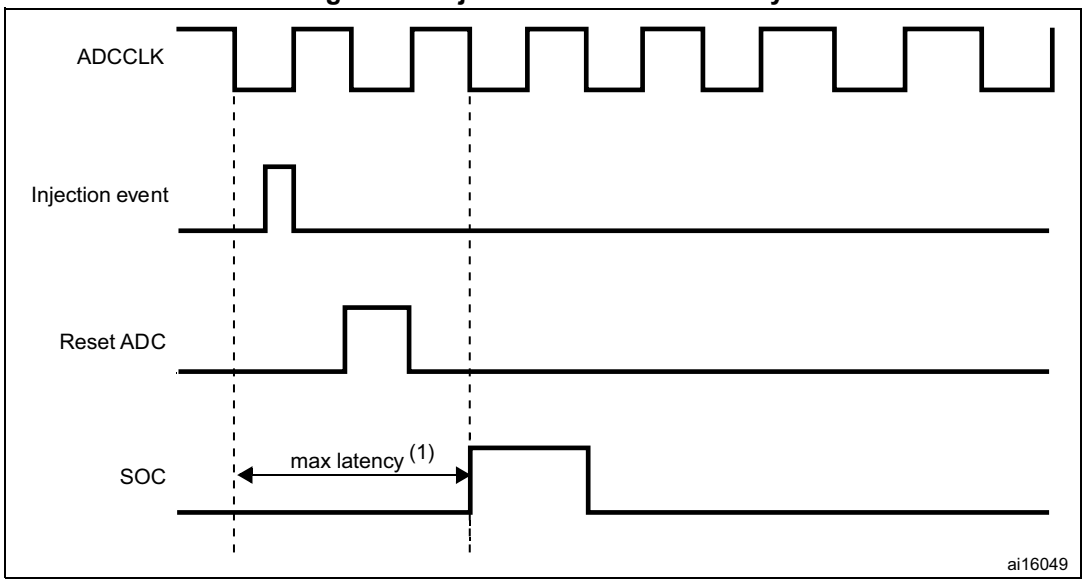
If the JAUTO bit is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRx and ADC_JSQR registers.

In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

Note: It is not possible to use both the auto-injected and discontinuous modes simultaneously.

Figure 47. Injected conversion latency



1. The maximum latency value can be found in the electrical characteristics of the STM32F40x and STM32F41x datasheets.

13.3.10 Discontinuous mode

Regular group

This mode is enabled by setting the DISCEN bit in the ADC_CR1 register. It can be used to convert a short sequence of n conversions (n ≤8) that is part of the sequence of conversions selected in the ADC_SQRx registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CR1 register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRx registers until all the conversions in the sequence are done. The total sequence length is defined by the L[3:0] bits in the ADC_SQR1 register.

Example:

- n = 3, channels to be converted = 0, 1, 2, 3, 6, 7, 9, 10
- 1st trigger: sequence converted 0, 1, 2. An EOC event is generated at each conversion.
- 2nd trigger: sequence converted 3, 6, 7. An EOC event is generated at each conversion
- 3rd trigger: sequence converted 9, 10.An EOC event is generated at each conversion
- 4th trigger: sequence converted 0, 1, 2. An EOC event is generated at each conversion

Note: When a regular group is converted in discontinuous mode, no rollover occurs.

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the 4th trigger reconverts the channels 0, 1 and 2 in the 1st subgroup.

Injected group

This mode is enabled by setting the JDISCEN bit in the ADC_CR1 register. It can be used to convert the sequence selected in the ADC_JSQR register, channel by channel, after an external trigger event.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADC_JSQR register.

Example:

n = 1, channels to be converted = 1, 2, 3
1st trigger: channel 1 converted
2nd trigger: channel 2 converted
3rd trigger: channel 3 converted and JEOC event generated
4th trigger: channel 1

Note: When all injected channels are converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel 1.

It is not possible to use both the auto-injected and discontinuous modes simultaneously.

Discontinuous mode must not be set for regular and injected groups at the same time. Discontinuous mode must be enabled only for the conversion of one group.

13.4 Data alignment

The ALIGN bit in the ADC_CR2 register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in *Figure 48* and *Figure 49*.

The converted data value from the injected group of channels is decreased by the user-defined offset written in the ADC_JOFRx registers so the result can be a negative value. The SEXT bit represents the extended sign value.

For channels in a regular group, no offset is subtracted so only twelve bits are significant.

Figure 48. Right alignment of 12-bit data

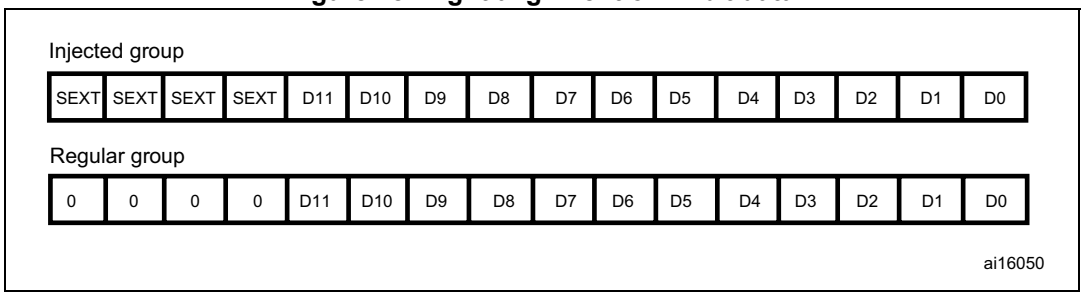
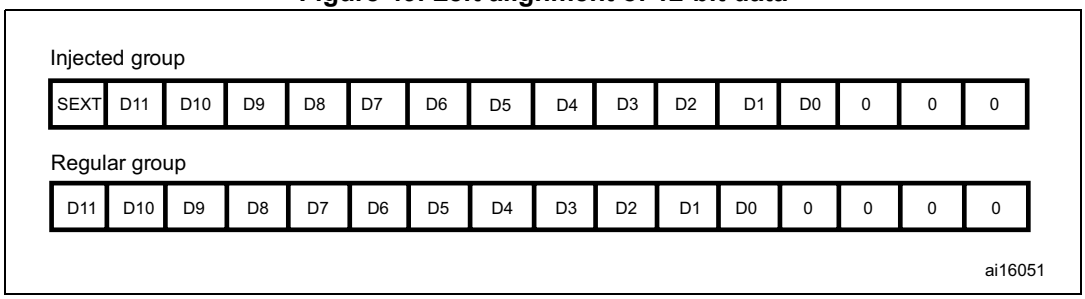
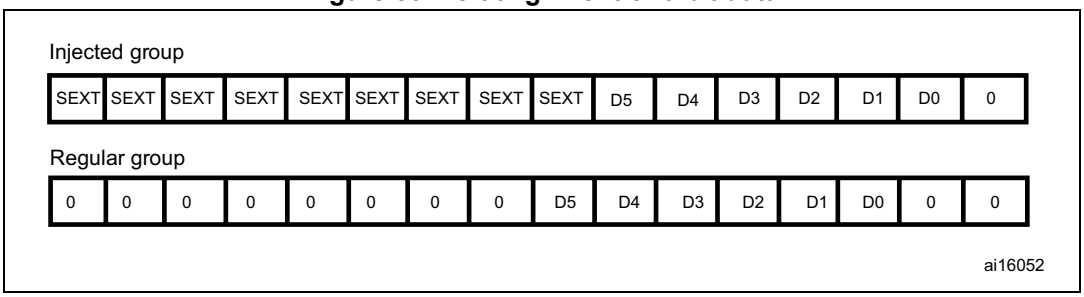


Figure 49. Left alignment of 12-bit data



Special case: when left-aligned, the data are aligned on a half-word basis except when the resolution is set to 6-bit. in that case, the data are aligned on a byte basis as shown in *Figure 50*.

Figure 50. Left alignment of 6-bit data



13.5 Channel-wise programmable sampling time

The ADC samples the input voltage for a number of ADCCLK cycles that can be modified using the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers. Each channel can be sampled with a different sampling time.

The total conversion time is calculated as follows:

T_{conv} = Sampling time + 12 cycles

Example:

With ADCCLK = 30 MHz and sampling time = 3 cycles:

T_{conv} = 3 + 12 = 15 cycles = 0.5 µs with APB2 at 60 MHz

13.6 Conversion on external trigger and trigger polarity

Conversion can be triggered by an external event (e.g. timer capture, EXTI line). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from “0b00”, then external events are able to trigger a conversion with the selected polarity. *Table 67* provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 67. Configuring the trigger polarity

Source	EXTEN[1:0] / JEXTEN[1:0]
Trigger detection disabled	00
Detection on the rising edge	01
Detection on the falling edge	10
Detection on both the rising and falling edges	11

Note: The polarity of the external trigger can be changed on the fly.

The EXTSEL[3:0] and JEXTSEL[3:0] control bits are used to select which out of 16 possible events can trigger conversion for the regular and injected groups.

Table 68 gives the possible external trigger for regular conversion.

Table 68. External trigger for regular channels

Source	Type	EXTSEL[3:0]
TIM1_CH1 event	Internal signal from on-chip timers	0000
TIM1_CH2 event		0001
TIM1_CH3 event		0010
TIM2_CH2 event		0011
TIM2_CH3 event		0100
TIM2_CH4 event		0101
TIM2_TRGO event		0110
TIM3_CH1 event		0111
TIM3_TRGO event		1000
TIM4_CH4 event		1001
TIM5_CH1 event		1010
TIM5_CH2 event		1011
TIM5_CH3 event		1100
TIM8_CH1 event		1101
TIM8_TRGO event		1110
EXTI line11	External pin	1111

Table 69 gives the possible external trigger for injected conversion.

Table 69. External trigger for injected channels

Source	Connection type	JEXTSEL[3:0]
TIM1_CH4 event	Internal signal from on-chip timers	0000
TIM1_TRGO event		0001
TIM2_CH1 event		0010
TIM2_TRGO event		0011
TIM3_CH2 event		0100
TIM3_CH4 event		0101
TIM4_CH1 event		0110
TIM4_CH2 event		0111
TIM4_CH3 event		1000
TIM4_TRGO event		1001
TIM5_CH4 event		1010
TIM5_TRGO event		1011
TIM8_CH2 event		1100
TIM8_CH3 event		1101
TIM8_CH4 event		1110
EXTI line15	External pin	1111

Software source trigger events can be generated by setting SWSTART (for regular conversion) or JSWSTART (for injected conversion) in ADC_CR2.

A regular group conversion can be interrupted by an injected trigger.

Note: The trigger selection can be changed on the fly. However, when the selection changes, there is a time frame of 1 APB clock cycle during which the trigger detection is disabled. This is to avoid spurious detection during transitions.

13.7 Fast conversion mode

It is possible to perform faster conversion by reducing the ADC resolution. The RES bits are used to select the number of bits available in the data register. The minimum conversion time for each resolution is then as follows:

- 12 bits: 3 + 12 = 15 ADCCLK cycles
- 10 bits: 3 + 10 = 13 ADCCLK cycles
- 8 bits: 3 + 8 = 11 ADCCLK cycles
- 6 bits: 3 + 6 = 9 ADCCLK cycles

13.8 Data management

13.8.1 Using the DMA

Since converted regular channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one regular channel. This avoids the loss of the data already stored in the ADC_DR register.

When the DMA mode is enabled (DMA bit set to 1 in the ADC_CR2 register), after each conversion of a regular channel, a DMA request is generated. This allows the transfer of the converted data from the ADC_DR register to the destination location selected by the software.

Despite this, if data are lost (overrun), the OVR bit in the ADC_SR register is set and an interrupt is generated (if the OVRIE enable bit is set). DMA transfers are then disabled and DMA requests are no longer accepted. In this case, if a DMA request is made, the regular conversion in progress is aborted and further regular triggers are ignored. It is then necessary to clear the OVR flag and the DMAEN bit in the used DMA stream, and to re-initialize both the DMA and the ADC to have the wanted converted channel data transferred to the right memory location. Only then can the conversion be resumed and the data transfer, enabled again. Injected channel conversions are not impacted by overrun errors.

When OVR = 1 in DMA mode, the DMA requests are blocked after the last valid data have been transferred, which means that all the data transferred to the RAM can be considered as valid.

At the end of the last DMA transfer (number of transfers configured in the DMA controller's DMA_SxNTR register):

- No new DMA request is issued to the DMA controller if the DDS bit is cleared to 0 in the ADC_CR2 register (this avoids generating an overrun error). However the DMA bit is not cleared by hardware. It must be written to 0, then to 1 to start a new transfer.
- Requests can continue to be generated if the DDS bit is set to 1. This allows configuring the DMA in double-buffer circular mode.

To recover the ADC from OVR state when the DMA is used, follow the steps below:

1. Reinitialize the DMA (adjust destination address and NDTR counter)
2. Clear the ADC OVR bit in ADC_SR register
3. Trigger the ADC to start the conversion.

13.8.2 Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the EOCS bit must be set in the ADC_CR2 register for the EOC status bit to be set at the end of each conversion, and not only at the end of the sequence. When EOCS = 1, overrun detection is automatically enabled. Thus, each time a conversion is complete, EOC is set and the ADC_DR register can be read. The overrun management is the same as when the DMA is used.

To recover the ADC from OVR state when the EOCS is set, follow the steps below:

1. Clear the ADC OVR bit in ADC_SR register
2. Trigger the ADC to start the conversion.

13.8.3 Conversions without DMA and without overrun detection

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). For that, the DMA must be disabled (DMA = 0) and the EOC bit must be set at the end of a sequence only (EOCS = 0). In this configuration, overrun detection is disabled.

13.9 Multi ADC mode

In devices with two ADCs or more, the Dual (with two ADCs) and Triple (with three ADCs) ADC modes can be used (see *Figure 51*).

In multi ADC mode, the start of conversion is triggered alternately or simultaneously by the ADC1 master to the ADC2 and ADC3 slaves, depending on the mode selected by the MULTI[4:0] bits in the ADC_CCR register.

Note: In multi ADC mode, when configuring conversion trigger by an external event, the application must set trigger by the master only and disable trigger by slaves to prevent spurious triggers that would start unwanted slave conversions.

The four possible modes below are implemented:

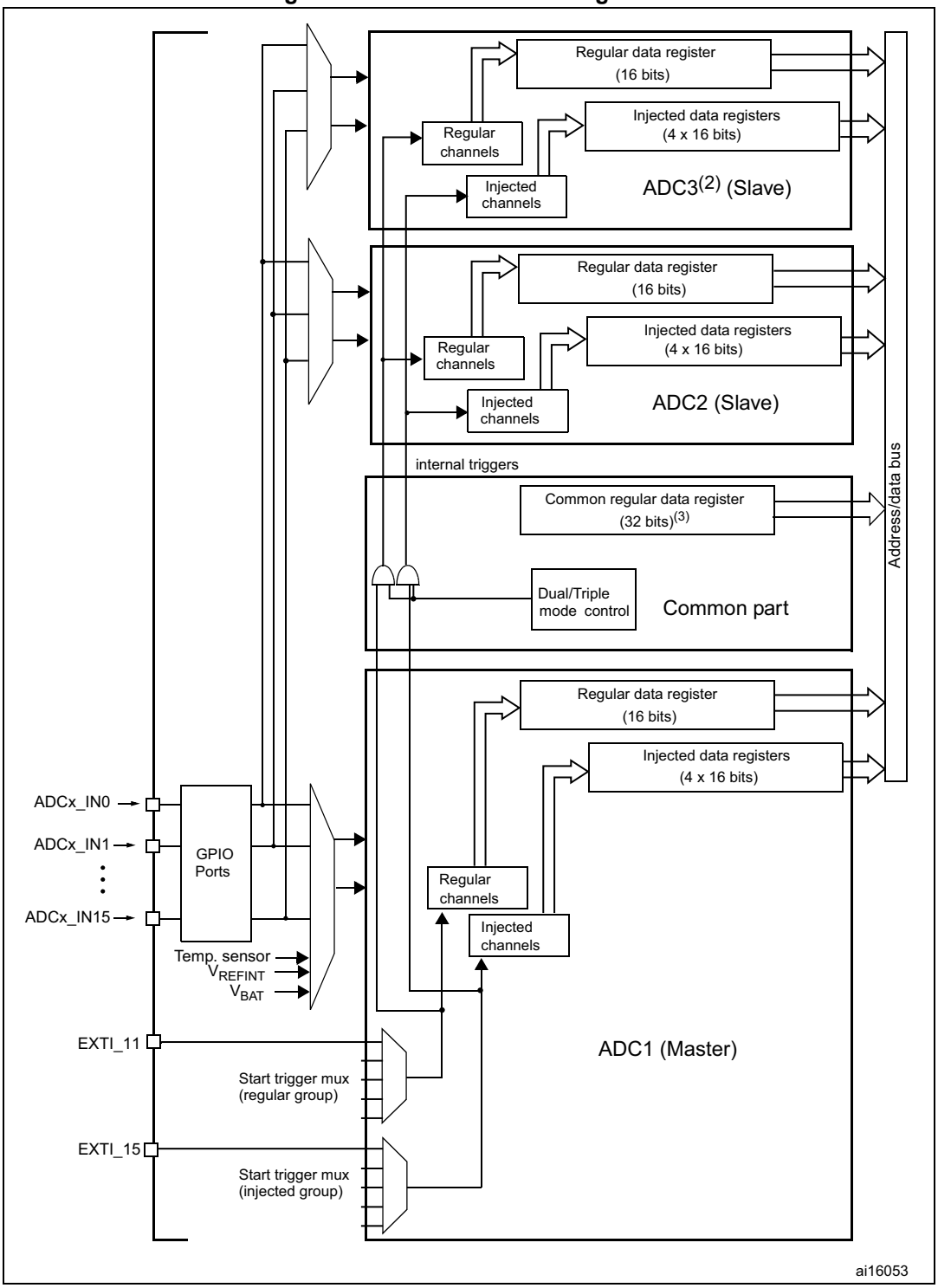
- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use the previous modes combined in the following ways:

- Injected simultaneous mode + Regular simultaneous mode
- Regular simultaneous mode + Alternate trigger mode

Note: In multi ADC mode, the converted data can be read on the multi-mode data register (ADC_CDR). The status bits can be read in the multi-mode status register (ADC_CSR).

Figure 51. Multi ADC block diagram(1)



1. Although external triggers are present on ADC2 and ADC3 they are not shown in this diagram.
2. In the Dual ADC mode, the ADC3 slave part is not present.
3. In Triple ADC mode, the ADC common data register (ADC_CDR) contains the ADC1, ADC2 and ADC3's regular converted data. All 32 register bits are used according to a selected storage order.
In Dual ADC mode, the ADC common data register (ADC_CDR) contains both the ADC1 and ADC2's regular converted data. All 32 register bits are used.

- DMA requests in Multi ADC mode:

  In Multi ADC mode the DMA may be configured to transfer converted data in three different modes. In all cases, the DMA streams to use are those connected to the ADC:

  – **DMA mode 1:** On each DMA request (one data item is available), a half-word representing an ADC-converted data item is transferred.

    In Triple ADC mode, ADC1 data are transferred on the first request, ADC2 data are transferred on the second request and ADC3 data are transferred on the third request; the sequence is repeated. So the DMA first transfers ADC1 data followed by ADC2 data followed by ADC3 data and so on.

    DMA mode 1 is used in regular simultaneous triple mode only.

    Example:

    Regular simultaneous triple mode: 3 consecutive DMA requests are generated (one for each converted data item)

    1st request: ADC_CDR[31:0] = ADC1_DR[15:0]

    2nd request: ADC_CDR[31:0] = ADC2_DR[15:0]

    3rd request: ADC_CDR[31:0] = ADC3_DR[15:0]

    4th request: ADC_CDR[31:0] = ADC1_DR[15:0]

  – **DMA mode 2**: On each DMA request (two data items are available) two half-words representing two ADC-converted data items are transferred as a word.

    In Dual ADC mode, both ADC2 and ADC1 data are transferred on the first request (ADC2 data take the upper half-word and ADC1 data take the lower half-word) and so on.

    In Triple ADC mode, three DMA requests are generated. On the first request, both ADC2 and ADC1 data are transferred (ADC2 data take the upper half-word and ADC1 data take the lower half-word). On the second request, both ADC1 and ADC3 data are transferred (ADC1 data take the upper half-word and ADC3 data take the lower half-word).On the third request, both ADC3 and ADC2 data are transferred (ADC3 data take the upper half-word and ADC2 data take the lower half-word) and so on.

    DMA mode 2 is used in interleaved mode and in regular simultaneous mode (for Dual ADC mode only).

    Example:

    a) Interleaved dual mode: a DMA request is generated each time 2 data items are available:

    1st request: ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0]

    2nd request: ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0]

    b) Interleaved triple mode: a DMA request is generated each time 2 data items are available

    1st request: ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0]

    2nd request: ADC_CDR[31:0] = ADC1_DR[15:0] | ADC3_DR[15:0]

    3rd request: ADC_CDR[31:0] = ADC3_DR[15:0] | ADC2_DR[15:0]

    4th request: ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0]

  – **DMA mode 3**: This mode is similar to the DMA mode 2. The only differences are that the on each DMA request (two data items are available) two bytes

representing two ADC converted data items are transferred as a half-word. The data transfer order is similar to that of the DMA mode 2.

DMA mode 3 is used in interleaved mode in 6-bit and 8-bit resolutions (dual and triple mode).

Example:

a) Interleaved dual mode: a DMA request is generated each time 2 data items are available

1st request: ADC_CDR[15:0] = ADC2_DR[7:0] | ADC1_DR[7:0]

2nd request: ADC_CDR[15:0] = ADC2_DR[7:0] | ADC1_DR[7:0]

b) Interleaved triple mode: a DMA request is generated each time 2 data items are available

1st request: ADC_CDR[15:0] = ADC2_DR[7:0] | ADC1_DR[7:0]

2nd request: ADC_CDR[15:0] = ADC1_DR[7:0] | ADC3_DR[7:0]

3rd request: ADC_CDR[15:0] = ADC3_DR[7:0] | ADC2_DR[7:0]

4th request: ADC_CDR[15:0] = ADC2_DR[7:0] | ADC1_DR[7:0]

Overrun detection: If an overrun is detected on one of the concerned ADCs (ADC1 and ADC2 in dual and triple modes, ADC3 in triple mode only), the DMA requests are no longer issued to ensure that all the data transferred to the RAM are valid. It may happen that the EOC bit corresponding to one ADC remains set because the data register of this ADC contains valid data.

13.9.1 Injected simultaneous mode

This mode converts an injected group of channels. The external trigger source comes from the injected group multiplexer of ADC1 (selected by the JEXTSEL[3:0] bits in the ADC1_CR2 register). A simultaneous trigger is provided to ADC2 and ADC3.

Note: Do not convert the same channel on the two/three ADCs (no overlapping sampling times for the two/three ADCs when converting the same channel).

In simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the longer of the 2 sequences (Dual ADC mode) /3 sequences (Triple ADC mode). Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

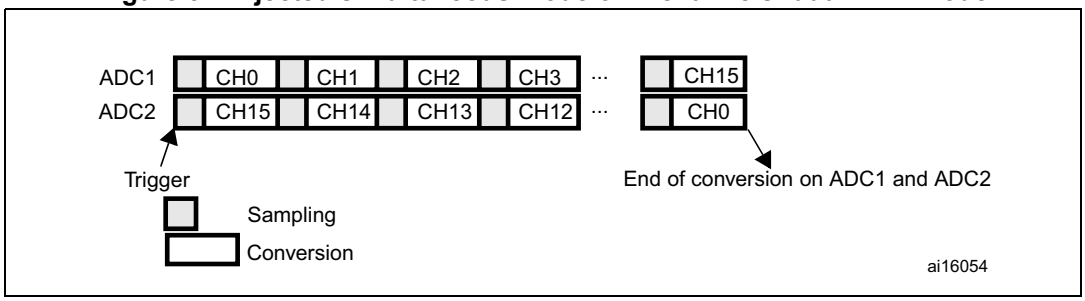
Regular conversions can be performed on one or all ADCs. In that case, they are independent of each other and are interrupted when an injected event occurs. They are resumed at the end of the injected conversion group.

Dual ADC mode

At the end of conversion event on ADC1 or ADC2:

- The converted data are stored into the ADC_JDRx registers of each ADC interface.
- A JEOC interrupt is generated (if enabled on one of the two ADC interfaces) when the ADC1/ADC2's injected channels have all been converted.

Figure 52. Injected simultaneous mode on 4 channels: dual ADC mode

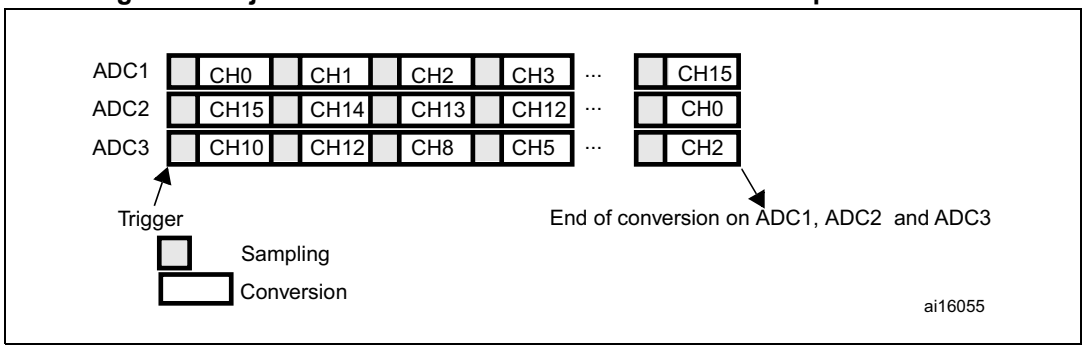


Triple ADC mode

At the end of conversion event on ADC1, ADC2 or ADC3:

- The converted data are stored into the ADC_JDRx registers of each ADC interface.
- A JEOC interrupt is generated (if enabled on one of the three ADC interfaces) when the ADC1/ADC2/ADC3's injected channels have all been converted.

Figure 53. Injected simultaneous mode on 4 channels: triple ADC mode



13.9.2 Regular simultaneous mode

This mode is performed on a regular group of channels. The external trigger source comes from the regular group multiplexer of ADC1 (selected by the EXTSEL[3:0] bits in the ADC1_CR2 register). A simultaneous trigger is provided to ADC2 and ADC3.

Note: Do not convert the same channel on the two/three ADCs (no overlapping sampling times for the two/three ADCs when converting the same channel).

In regular simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences (Dual ADC mode) /3 sequences (Triple ADC mode). Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

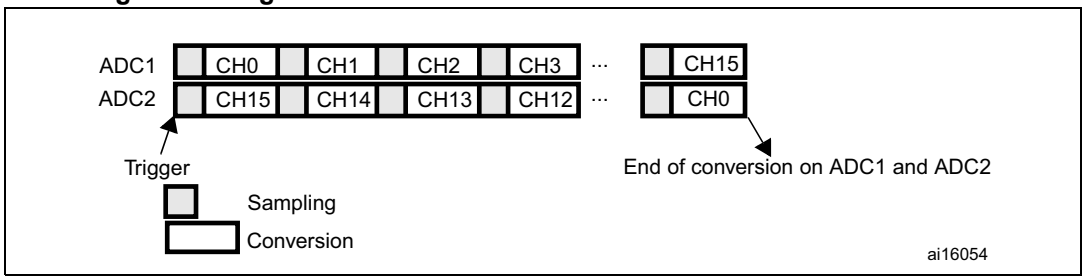
Injected conversions must be disabled.

Dual ADC mode

At the end of conversion event on ADC1 or ADC2:

- A 32-bit DMA transfer request is generated (if DMA[1:0] bits in the ADC_CCR register are equal to 0b10). This request transfers the ADC2 converted data stored in the upper half-word of the ADC_CDR 32-bit register to the SRAM and then the ADC1 converted data stored in the lower half-word of ADC_CDR to the SRAM.
- An EOC interrupt is generated (if enabled on one of the two ADC interfaces) when the ADC1/ADC2's regular channels have all been converted.

Figure 54. Regular simultaneous mode on 16 channels: dual ADC mode

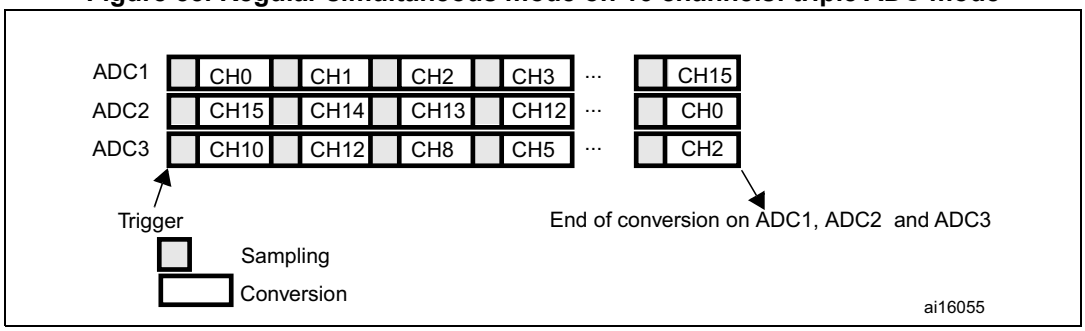


Triple ADC mode

At the end of conversion event on ADC1, ADC2 or ADC3:

- Three 32-bit DMA transfer requests are generated (if DMA[1:0] bits in the ADC_CCR register are equal to 0b01). Three transfers then take place from the ADC_CDR 32-bit register to SRAM: first the ADC1 converted data, then the ADC2 converted data and finally the ADC3 converted data. The process is repeated for each new three conversions.
- An EOC interrupt is generated (if enabled on one of the three ADC interfaces) when the ADC1/ADC2/ADC3's regular channels are have all been converted.

Figure 55. Regular simultaneous mode on 16 channels: triple ADC mode



13.9.3 Interleaved mode

This mode can be started only on a regular group (usually one channel). The external trigger source comes from the regular channel multiplexer of ADC1.

Dual ADC mode

After an external trigger occurs:

- ADC1 starts immediately
- ADC2 starts after a delay of several ADC clock cycles

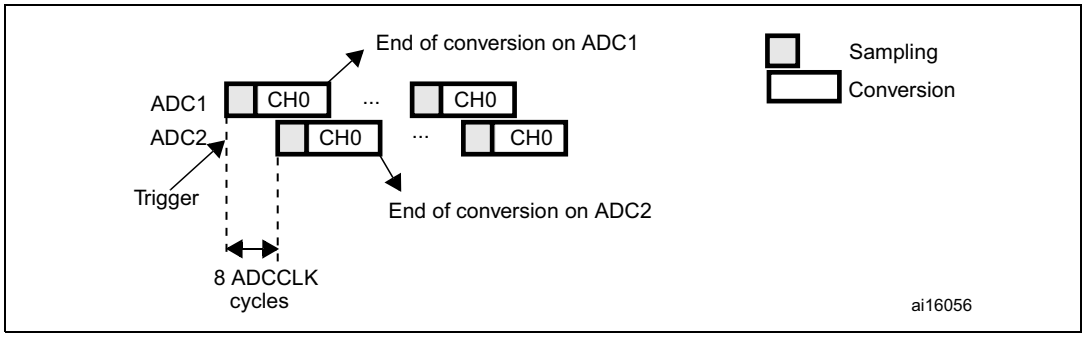
The minimum delay which separates 2 conversions in interleaved mode is configured in the DELAY bits in the ADC_CCR register. However, an ADC cannot start a conversion if the complementary ADC is still sampling its input (only one ADC can sample the input signal at a given time). In this case, the delay becomes the sampling time + 2 ADC clock cycles. For instance, if DELAY = 5 clock cycles and the sampling takes 15 clock cycles on both ADCs, then 17 clock cycles will separate conversions on ADC1 and ADC2).

If the CONT bit is set on both ADC1 and ADC2, the selected regular channels of both ADCs are continuously converted.

Note: If the conversion sequence is interrupted (for instance when DMA end of transfer occurs), the multi-ADC sequencer must be reset by configuring it in independent mode first (bits DUAL[4:0] = 00000) before reprogramming the interleaved mode.

After an EOC interrupt is generated by ADC2 (if enabled through the EOCIE bit) a 32-bit DMA transfer request is generated (if the DMA[1:0] bits in ADC_CCR are equal to 0b10). This request first transfers the ADC2 converted data stored in the upper half-word of the ADC_CDR 32-bit register into SRAM, then the ADC1 converted data stored in the register's lower half-word into SRAM.

Figure 56. Interleaved mode on 1 channel in continuous conversion mode: dual ADC mode



Triple ADC mode

After an external trigger occurs:

- ADC1 starts immediately and
- ADC2 starts after a delay of several ADC clock cycles
- ADC3 starts after a delay of several ADC clock cycles referred to the ADC2 conversion

The minimum delay which separates 2 conversions in interleaved mode is configured in the DELAY bits in the ADC_CCR register. However, an ADC cannot start a conversion if the complementary ADC is still sampling its input (only one ADC can sample the input signal at

a given time). In this case, the delay becomes the sampling time + 2 ADC clock cycles. For instance, if DELAY = 5 clock cycles and the sampling takes 15 clock cycles on the three ADCs, then 17 clock cycles will separate the conversions on ADC1, ADC2 and ADC3).

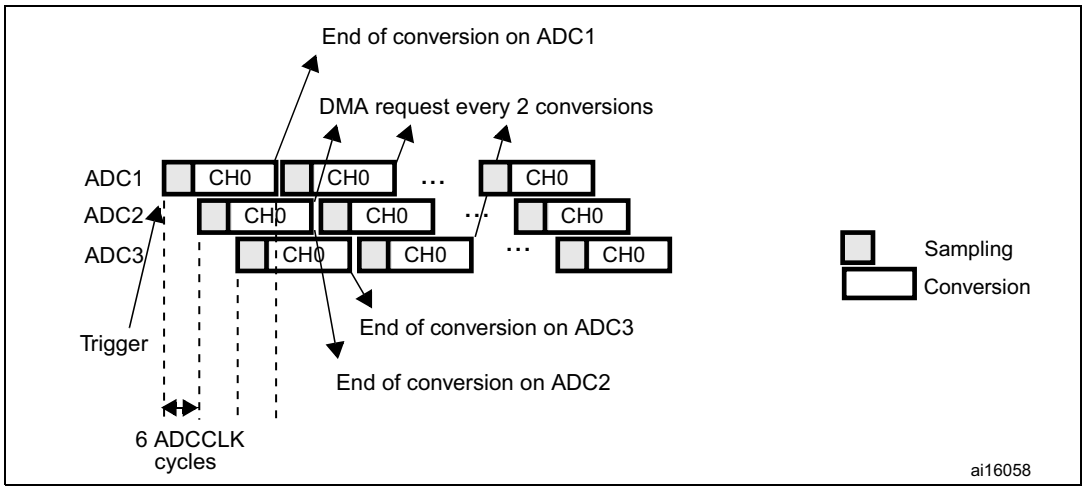
If the CONT bit is set on ADC1, ADC2 and ADC3, the selected regular channels of all ADCs are continuously converted.

Note: If the conversion sequence is interrupted (for instance when DMA end of transfer occurs), the multi-ADC sequencer must be reset by configuring it in independent mode first (bits DUAL[4:0] = 00000) before reprogramming the interleaved mode.

In this mode a DMA request is generated each time 2 data items are available, (if the DMA[1:0] bits in the ADC_CCR register are equal to 0b10). The request first transfers the first converted data stored in the lower half-word of the ADC_CDR 32-bit register to SRAM, then it transfers the second converted data stored in ADC_CDR's upper half-word to SRAM. The sequence is the following:

- 1st request: ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0]
- 2nd request: ADC_CDR[31:0] = ADC1_DR[15:0] | ADC3_DR[15:0]
- 3rd request: ADC_CDR[31:0] = ADC3_DR[15:0] | ADC2_DR[15:0]
- 4th request: ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0], ...

Figure 57. Interleaved mode on 1 channel in continuous conversion mode: triple ADC mode



13.9.4 Alternate trigger mode

This mode can be started only on an injected group. The source of external trigger comes from the injected group multiplexer of ADC1.

Note: Regular conversions can be enabled on one or all ADCs. In this case the regular conversions are independent of each other. A regular conversion is interrupted when the

ADC has to perform an injected conversion. It is resumed when the injected conversion is finished.

If the conversion sequence is interrupted (for instance when DMA end of transfer occurs), the multi-ADC sequencer must be reset by configuring it in independent mode first (bits DUAL[4:0] = 00000) before reprogramming the interleaved mode.

The time interval between 2 trigger events must be greater than or equal to 1 ADC clock period. The minimum time interval between 2 trigger events that start conversions on the same ADC is the same as in the single ADC mode.

Dual ADC mode

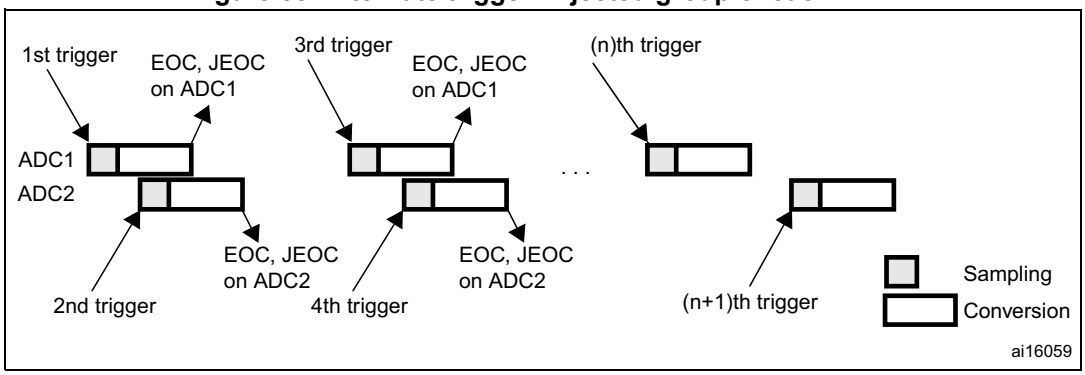
- When the 1st trigger occurs, all injected ADC1 channels in the group are converted
- When the 2nd trigger occurs, all injected ADC2 channels in the group are converted
- and so on

A JEOC interrupt, if enabled, is generated after all injected ADC1 channels in the group have been converted.

A JEOC interrupt, if enabled, is generated after all injected ADC2 channels in the group have been converted.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts by converting the injected ADC1 channels in the group.

Figure 58. Alternate trigger: injected group of each ADC



If the injected discontinuous mode is enabled for both ADC1 and ADC2:

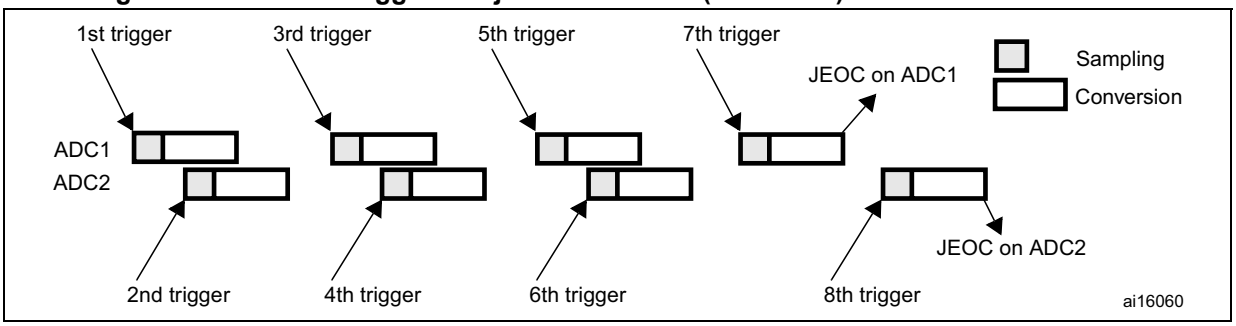
- When the 1st trigger occurs, the first injected ADC1 channel is converted.
- When the 2nd trigger occurs, the first injected ADC2 channel are converted
- and so on

A JEOC interrupt, if enabled, is generated after all injected ADC1 channels in the group have been converted.

A JEOC interrupt, if enabled, is generated after all injected ADC2 channels in the group have been converted.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts.

Figure 59. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode



Triple ADC mode

- When the 1st trigger occurs, all injected ADC1 channels in the group are converted.
- When the 2nd trigger occurs, all injected ADC2 channels in the group are converted.
- When the 3rd trigger occurs, all injected ADC3 channels in the group are converted.
- and so on

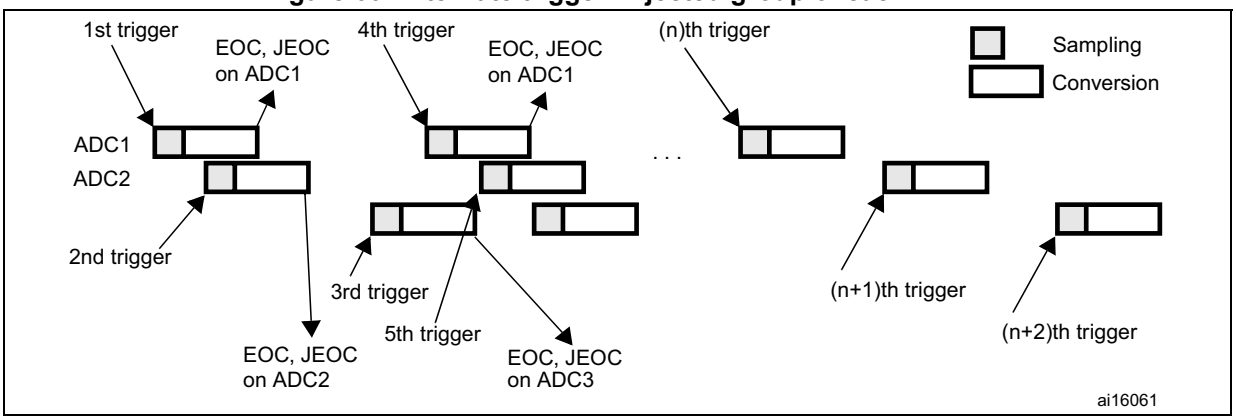
A JEOC interrupt, if enabled, is generated after all injected ADC1 channels in the group have been converted.

A JEOC interrupt, if enabled, is generated after all injected ADC2 channels in the group have been converted.

A JEOC interrupt, if enabled, is generated after all injected ADC3 channels in the group have been converted.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts by converting the injected ADC1 channels in the group.

Figure 60. Alternate trigger: injected group of each ADC



13.9.5 Combined regular/injected simultaneous mode

It is possible to interrupt the simultaneous conversion of a regular group to start the simultaneous conversion of an injected group.

Note: In combined regular/injected simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences (Dual ADC mode) /3 sequences (Triple ADC mode). Otherwise, the

ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

13.9.6 Combined regular simultaneous + alternate trigger mode

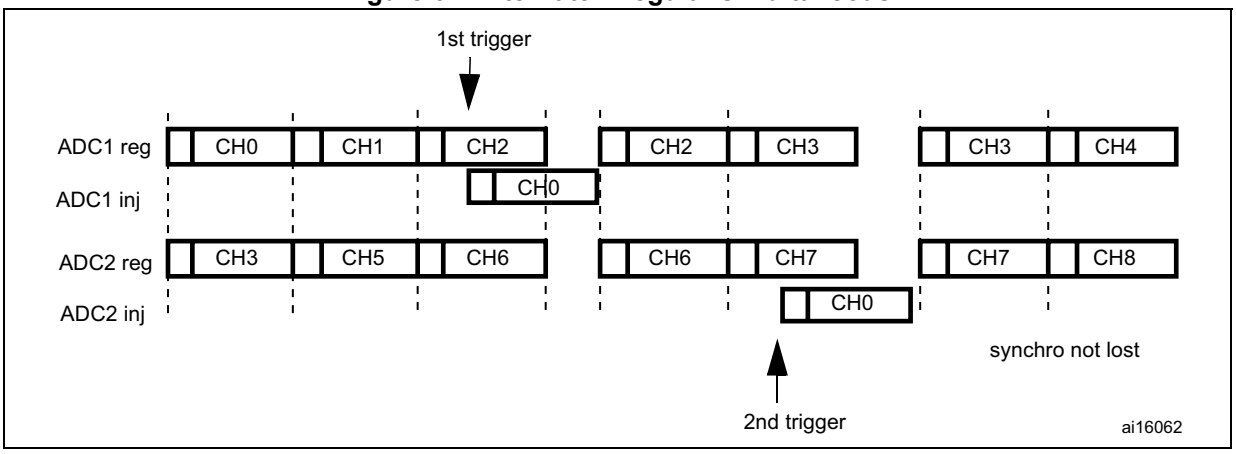
It is possible to interrupt the simultaneous conversion of a regular group to start the alternate trigger conversion of an injected group. *Figure 61* shows the behavior of an alternate trigger interrupting a simultaneous regular conversion.

The injected alternate conversion is immediately started after the injected event. If regular conversion is already running, in order to ensure synchronization after the injected conversion, the regular conversion of all (master/slave) ADCs is stopped and resumed synchronously at the end of the injected conversion.

Note: In combined regular simultaneous + alternate trigger mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences (Dual ADC mode) /3 sequences (Triple ADC mode). Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

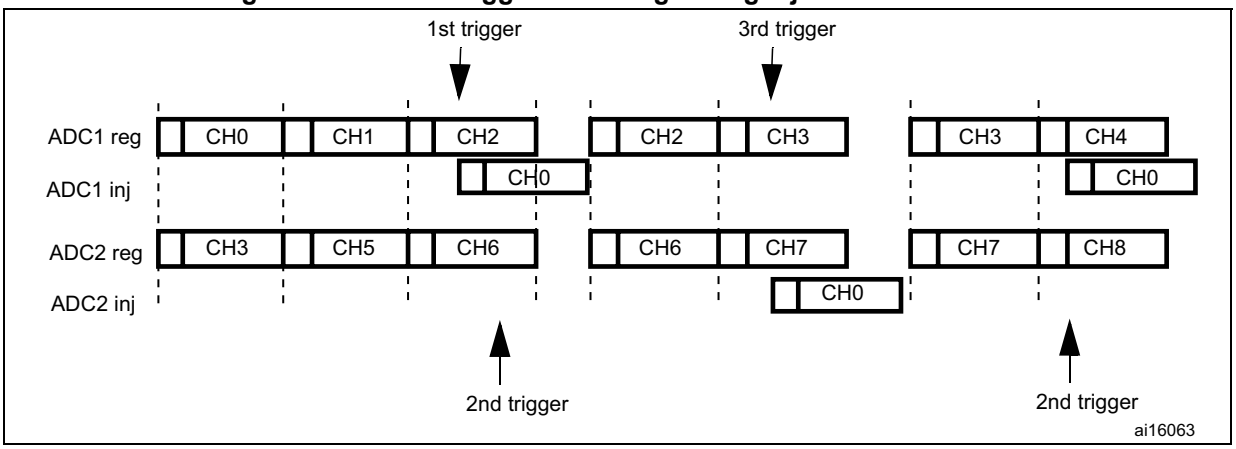
If the conversion sequence is interrupted (for instance when DMA end of transfer occurs), the multi-ADC sequencer must be reset by configuring it in independent mode first (bits DUAL[4:0] = 00000) before reprogramming the interleaved mode.

Figure 61. Alternate + regular simultaneous



If a trigger occurs during an injected conversion that has interrupted a regular conversion, it is ignored. *Figure 62* shows the behavior in this case (2nd trigger is ignored).

Figure 62. Case of trigger occurring during injected conversion



13.10 Temperature sensor

The temperature sensor can be used to measure the ambient temperature (T_A) of the device.

- On STM32F40x and STM32F41x devices, the temperature sensor is internally connected to ADC1_IN16 channel which is used to convert the sensor output voltage to a digital value.
- On STM32F42x and STM32F43x devices, the temperature sensor is internally connected to the same input channel, ADC1_IN18, as VBAT: ADC1_IN18 is used to convert the sensor output voltage or VBAT into a digital value. Only one conversion, temperature sensor or VBAT, must be selected at a time. When the temperature sensor and the VBAT conversion are set simultaneously, only the VBAT conversion is performed.

Figure 63 shows the block diagram of the temperature sensor.

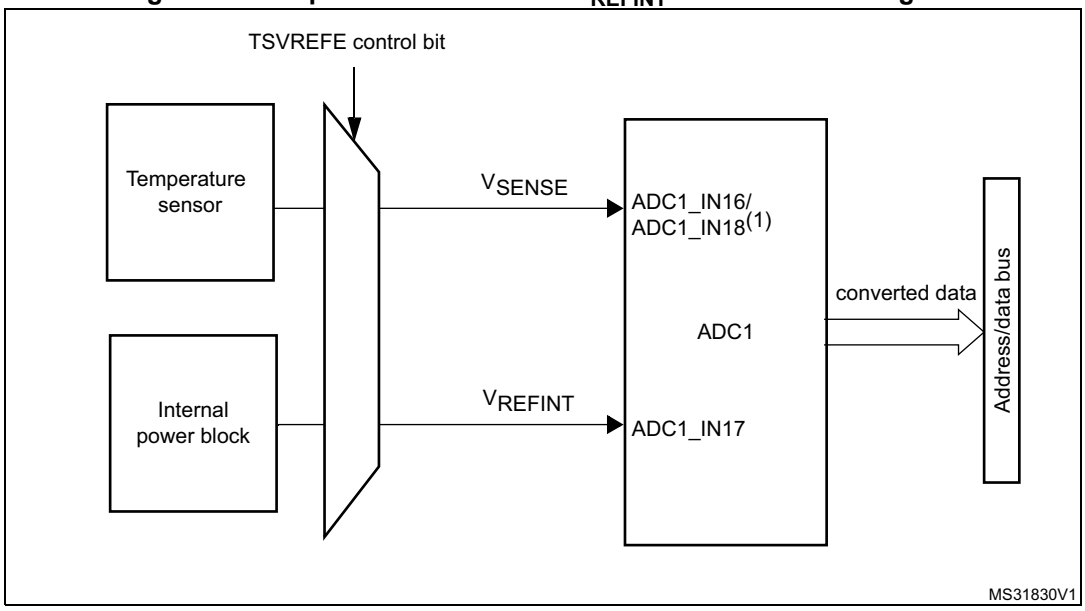
When not in use, the sensor can be put in power down mode.

Note: The TSVREFE bit must be set to enable the conversion of both internal channels: the ADC1_IN16 or ADC1_IN18 (temperature sensor) and the ADC1_IN17 (VREFINT).

Main features

- Supported temperature range: –40 to 125 °C
- Precision: ±1.5 °C

Figure 63. Temperature sensor and V_{REFINT} channel block diagram



1. V_{SENSE} is input to ADC1_IN16 for the STM23F40x and STM32F41x devices and to ADC1_IN18 for the STM32F42x and STM32F43x devices.

Reading the temperature

To use the sensor:

3. Select ADC1_IN16 or ADC1_IN18 input channel.
4. Select a sampling time greater than the minimum sampling time specified in the datasheet.
5. Set the TSVREFE bit in the ADC_CCR register to wake up the temperature sensor from power down mode
6. Start the ADC conversion by setting the SWSTART bit (or by external trigger)
7. Read the resulting V_{SENSE} data in the ADC data register
8. Calculate the temperature using the following formula:

   Temperature (in °C) = {(V_{SENSE} – V_{25}) / Avg_Slope} + 25

   Where:

   – V_{25} = V_{SENSE} value for 25° C
   – Avg_Slope = average slope of the temperature vs. V_{SENSE} curve (given in mV/°C or µV/°C)

   Refer to the datasheet's electrical characteristics section for the actual values of V_{25} and Avg_Slope.

Note: The sensor has a startup time after waking from power down mode before it can output V_{SENSE} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADON and TSVREFE bits should be set at the same time.

The temperature sensor output voltage changes linearly with temperature. The offset of this linear function depends on each chip due to process variation (up to 45 °C from one chip to another).

The internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. If accurate temperature reading is required, an external temperature sensor should be used.

13.11 Battery charge monitoring

The VBATE bit in the ADC_CCR register is used to switch to the battery voltage. As the V_{BAT} voltage could be higher than V_{DDA}, to ensure the correct operation of the ADC, the V_{BAT} pin is internally connected to a bridge divider.

When the VBATE is set, the bridge is automatically enabled to connect:

- VBAT/2 to the ADC1_IN18 input channel, on STM32F40xx and STM32F41xx devices
- VBAT/4 to the ADC1_IN18 input channel, on STM32F42xx and STM32F43xx devices

Note: On STM32F42xx and STM32F43xx devices, VBAT and temperature sensor are connected to the same ADC internal channel (ADC1_IN18). Only one conversion, either temperature sensor or VBAT, must be selected at a time. When both conversion are enabled simultaneously, only the VBAT conversion is performed.

13.12 ADC interrupts

An interrupt can be produced on the end of conversion for regular and injected groups, when the analog watchdog status bit is set and when the overrun status bit is set. Separate interrupt enable bits are available for flexibility.

Two other flags are present in the ADC_SR register, but there is no interrupt associated with them:

- JSTRT (Start of conversion for channels of an injected group)
- STRT (Start of conversion for channels of a regular group)

Table 70. ADC interrupts

Interrupt event	Event flag	Enable control bit
End of conversion of a regular group	EOC	EOCIE
End of conversion of an injected group	JEOC	JEOCIE
Analog watchdog status bit is set	AWD	AWDIE
Overrun	OVR	OVRIE

13.13 ADC registers

Refer to *Section 1.1: List of abbreviations for registers* for registers for a list of abbreviations used in register descriptions.

The peripheral registers must be written at word level (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

13.13.1 ADC status register (ADC_SR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										OVR	STRT	JSTRT	JEOC	EOC	AWD
										rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

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

Bit 5 **OVR:** Overrun

This bit is set by hardware when data are lost (either in single mode or in dual/triple mode). It is cleared by software. Overrun detection is enabled only when DMA = 1 or EOCS = 1.

0: No overrun occurred

1: Overrun has occurred

Bit 4 **STRT:** Regular channel start flag

This bit is set by hardware when regular channel conversion starts. It is cleared by software.

0: No regular channel conversion started

1: Regular channel conversion has started

Bit 3 **JSTRT:** Injected channel start flag

This bit is set by hardware when injected group conversion starts. It is cleared by software.

0: No injected group conversion started

1: Injected group conversion has started

Bit 2 **JEOC:** Injected channel end of conversion

This bit is set by hardware at the end of the conversion of all injected channels in the group. It is cleared by software.

0: Conversion is not complete

1: Conversion complete

Bit 1 **EOC:** Regular channel end of conversion

This bit is set by hardware at the end of the conversion of a regular group of channels. It is cleared by software or by reading the ADC_DR register.

0: Conversion not complete (EOCS=0), or sequence of conversions not complete (EOCS=1)

1: Conversion complete (EOCS=0), or sequence of conversions complete (EOCS=1)

Bit 0 **AWD:** Analog watchdog flag

This bit is set by hardware when the converted voltage crosses the values programmed in the ADC_LTR and ADC_HTR registers. It is cleared by software.

0: No analog watchdog event occurred

1: Analog watchdog event occurred

13.13.2 ADC control register 1 (ADC_CR1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					OVRIE	RES		AWDEN	JAWDEN	Reserved					
					rw	rw	rw	rw	rw						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 26 **OVRIE:** Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt.
0: Overrun interrupt disabled
1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

Bits 25:24 **RES[1:0]:** Resolution

These bits are written by software to select the resolution of the conversion.
00: 12-bit (15 ADCCLK cycles)
01: 10-bit (13 ADCCLK cycles)
10: 8-bit (11 ADCCLK cycles)
11: 6-bit (9 ADCCLK cycles)

Bit 23 **AWDEN:** Analog watchdog enable on regular channels

This bit is set and cleared by software.
0: Analog watchdog disabled on regular channels
1: Analog watchdog enabled on regular channels

Bit 22 **JAWDEN:** Analog watchdog enable on injected channels

This bit is set and cleared by software.
0: Analog watchdog disabled on injected channels
1: Analog watchdog enabled on injected channels

Bits 21:16 Reserved, must be kept at reset value.

Bits 15:13 **DISCNUM[2:0]:** Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.
000: 1 channel
001: 2 channels
...
111: 8 channels

Bit 12 **JDISCEN:** Discontinuous mode on injected channels

This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.
0: Discontinuous mode on injected channels disabled
1: Discontinuous mode on injected channels enabled

Bit 11 **DISCEN:** Discontinuous mode on regular channels

This bit is set and cleared by software to enable/disable Discontinuous mode on regular channels.

0: Discontinuous mode on regular channels disabled

1: Discontinuous mode on regular channels enabled

Bit 10 **JAUTO:** Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled

1: Automatic injected group conversion enabled

Bit 9 **AWDSGL:** Enable the watchdog on a single channel in scan mode

This bit is set and cleared by software to enable/disable the analog watchdog on the channel identified by the AWDCH[4:0] bits.

0: Analog watchdog enabled on all channels

1: Analog watchdog enabled on a single channel

Bit 8 **SCAN:** Scan mode

This bit is set and cleared by software to enable/disable the Scan mode. In Scan mode, the inputs selected through the ADC_SQRx or ADC_JSQRx registers are converted.

0: Scan mode disabled

1: Scan mode enabled

Note: An EOC interrupt is generated if the EOCIE bit is set:

- *At the end of each regular group sequence if the EOCS bit is cleared to 0*
- *At the end of each regular channel conversion if the EOCS bit is set to 1*

Note: A JEOC interrupt is generated only on the end of conversion of the last channel if the JEOCIE bit is set.

Bit 7 **JEOCIE:** Interrupt enable for injected channels

This bit is set and cleared by software to enable/disable the end of conversion interrupt for injected channels.

0: JEOC interrupt disabled

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Bit 6 **AWDIE:** Analog watchdog interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

Bit 5 **EOCIE:** Interrupt enable for EOC

This bit is set and cleared by software to enable/disable the end of conversion interrupt.

0: EOC interrupt disabled

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Bits 4:0 **AWDCH[4:0]:** Analog watchdog channel select bits

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

Note: 00000: ADC analog input Channel0
00001: ADC analog input Channel1
...
01111: ADC analog input Channel15
10000: ADC analog input Channel16
10001: ADC analog input Channel17
10010: ADC analog input Channel18
Other values reserved

13.13.3 ADC control register 2 (ADC_CR2)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved	SWSTART	EXTEN		EXTSEL[3:0]				reserved	JSWSTART	JEXTEN		JEXTSEL[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved				ALIGN	EOCS	DDS	DMA	Reserved						CONT	ADON
				rw	rw	rw	rw							rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SWSTART:** Start conversion of regular channels

This bit is set by software to start conversion and cleared by hardware as soon as the conversion starts.

0: Reset state

1: Starts conversion of regular channels

Note: This bit can be set only when ADON = 1 otherwise no conversion is launched.

Bits 29:28 **EXTEN:** External trigger enable for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Trigger detection disabled

01: Trigger detection on the rising edge

10: Trigger detection on the falling edge

11: Trigger detection on both the rising and falling edges

Bits 27:24 **EXTSEL[3:0]:** External event select for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

0000: Timer 1 CC1 event

0001: Timer 1 CC2 event

0010: Timer 1 CC3 event

0011: Timer 2 CC2 event

0100: Timer 2 CC3 event

0101: Timer 2 CC4 event

0110: Timer 2 TRGO event

0111: Timer 3 CC1 event

1000: Timer 3 TRGO event

1001: Timer 4 CC4 event

1010: Timer 5 CC1 event

1011: Timer 5 CC2 event

1100: Timer 5 CC3 event

1101: Timer 8 CC1 event

1110: Timer 8 TRGO event

1111: EXTI line11

Bit 23 Reserved, must be kept at reset value.

Bit 22 **JSWSTART:** Start conversion of injected channels

This bit is set by software and cleared by hardware as soon as the conversion starts.

0: Reset state

1: Starts conversion of injected channels

Note: This bit can be set only when ADON = 1 otherwise no conversion is launched.

Bits 21:20 **JEXTEN:** External trigger enable for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: Trigger detection disabled

01: Trigger detection on the rising edge

10: Trigger detection on the falling edge

11: Trigger detection on both the rising and falling edges

Bits 19:16 **JEXTSEL[3:0]:** External event select for injected group

These bits select the external event used to trigger the start of conversion of an injected group.

0000: Timer 1 CC4 event

0001: Timer 1 TRGO event

0010: Timer 2 CC1 event

0011: Timer 2 TRGO event

0100: Timer 3 CC2 event

0101: Timer 3 CC4 event

0110: Timer 4 CC1 event

0111: Timer 4 CC2 event

1000: Timer 4 CC3 event

1001: Timer 4 TRGO event

1010: Timer 5 CC4 event

1011: Timer 5 TRGO event

1100: Timer 8 CC2 event

1101: Timer 8 CC3 event

1110: Timer 8 CC4 event

1111: EXTI line15

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

Bit 11 **ALIGN:** Data alignment

This bit is set and cleared by software. Refer to *Figure 48* and *Figure 49*.

0: Right alignment

1: Left alignment

Bit 10 **EOCS:** End of conversion selection

This bit is set and cleared by software.

0: The EOC bit is set at the end of each sequence of regular conversions. Overrun detection is enabled only if DMA=1.

1: The EOC bit is set at the end of each regular conversion. Overrun detection is enabled.

Bit 9 **DDS:** DMA disable selection (for single ADC mode)

This bit is set and cleared by software.

0: No new DMA request is issued after the last transfer (as configured in the DMA controller)

1: DMA requests are issued as long as data are converted and DMA=1

Bit 8 **DMA:** Direct memory access mode (for single ADC mode)

This bit is set and cleared by software. Refer to the DMA controller chapter for more details.

0: DMA mode disabled

1: DMA mode enabled

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **CONT:** Continuous conversion

This bit is set and cleared by software. If it is set, conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

Bit 0 **ADON:** A/D Converter ON / OFF

This bit is set and cleared by software.

Note: 0: Disable ADC conversion and go to power down mode

1: Enable ADC

13.13.4 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
					rw	rw	rw	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.

Note: 000: 3 cycles
001: 15 cycles
010: 28 cycles
011: 56 cycles
100: 84 cycles
101: 112 cycles
110: 144 cycles
111: 480 cycles

13.13.5 ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP 5_0	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

Note: 000: 3 cycles
001: 15 cycles
010: 28 cycles
011: 56 cycles
100: 84 cycles
101: 112 cycles
110: 144 cycles
111: 480 cycles

13.13.6 ADC injected channel data offset register x (ADC_JOFRx) (x=1..4)

Address offset: 0x14-0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				JOFFSETx[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 **JOFFSETx[11:0]:** Data offset for injected channel x

These bits are written by software to define the offset to be subtracted from the raw converted data when converting injected channels. The conversion result can be read from in the ADC_JDRx registers.

13.13.7 ADC watchdog higher threshold register (ADC_HTR)

Address offset: 0x24

Reset value: 0x0000 0FFF

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **HT[11:0]:** Analog watchdog higher threshold

These bits are written by software to define the higher threshold for the analog watchdog.

Note: The software can write to these registers when an ADC conversion is ongoing. The programmed value will be effective when the next conversion is complete. Writing to this register is performed with a write delay that can create uncertainty on the effective time at which the new value is programmed.

13.13.8 ADC watchdog lower threshold register (ADC_LTR)

Address offset: 0x28

Reset value: 0x0000 0000

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **LT[11:0]:** Analog watchdog lower threshold

These bits are written by software to define the lower threshold for the analog watchdog.

Note: The software can write to these registers when an ADC conversion is ongoing. The programmed value will be effective when the next conversion is complete. Writing to this register is performed with a write delay that can create uncertainty on the effective time at which the new value is programmed.

13.13.9 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								L[3:0]				SQ16[4:1]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ16_0	SQ15[4:0]					SQ14[4:0]					SQ13[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

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

Bits 23:20 **L[3:0]:** Regular channel sequence length

These bits are written by software to define the total number of conversions in the regular channel conversion sequence.
0000: 1 conversion
0001: 2 conversions
...
1111: 16 conversions

Bits 19:15 **SQ16[4:0]:** 16th conversion in regular sequence

These bits are written by software with the channel number (0..18) assigned as the 16th in the conversion sequence.

Bits 14:10 **SQ15[4:0]:** 15th conversion in regular sequence

Bits 9:5 **SQ14[4:0]:** 14th conversion in regular sequence

Bits 4:0 **SQ13[4:0]:** 13th conversion in regular sequence

13.13.10 ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ12[4:0]					SQ11[4:0]					SQ10[4:1]			
		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
SQ10_0	SQ9[4:0]						SQ8[4:0]					SQ7[4:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:26 **SQ12[4:0]:** 12th conversion in regular sequence
These bits are written by software with the channel number (0..18) assigned as the 12th in the sequence to be converted.

Bits 24:20 **SQ11[4:0]:** 11th conversion in regular sequence

Bits 19:15 **SQ10[4:0]:** 10th conversion in regular sequence

Bits 14:10 **SQ9[4:0]:** 9th conversion in regular sequence

Bits 9:5 **SQ8[4:0]:** 8th conversion in regular sequence

Bits 4:0 **SQ7[4:0]:** 7th conversion in regular sequence

13.13.11 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ6[4:0]					SQ5[4:0]					SQ4[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4_0	SQ3[4:0]						SQ2[4:0]					SQ1[4:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:25 **SQ6[4:0]:** 6th conversion in regular sequence
These bits are written by software with the channel number (0..18) assigned as the 6th in the sequence to be converted.

Bits 24:20 **SQ5[4:0]:** 5th conversion in regular sequence

Bits 19:15 **SQ4[4:0]:** 4th conversion in regular sequence

Bits 14:10 **SQ3[4:0]:** 3rd conversion in regular sequence

Bits 9:5 **SQ2[4:0]:** 2nd conversion in regular sequence

Bits 4:0 **SQ1[4:0]:** 1st conversion in regular sequence

13.13.12 ADC injected sequence register (ADC_JSQR)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										JL[1:0]		JSQ4[4:1]			
										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ4[0]	JSQ3[4:0]					JSQ2[4:0]					JSQ1[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 21:20 **JL[1:0]:** Injected sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion
01: 2 conversions
10: 3 conversions
11: 4 conversions

Bits 19:15 **JSQ4[4:0]:** 4th conversion in injected sequence (when JL[1:0]=3, see note below)

These bits are written by software with the channel number (0..18) assigned as the 4th in the sequence to be converted.

Bits 14:10 **JSQ3[4:0]:** 3rd conversion in injected sequence (when JL[1:0]=3, see note below)

Bits 9:5 **JSQ2[4:0]:** 2nd conversion in injected sequence (when JL[1:0]=3, see note below)

Bits 4:0 **JSQ1[4:0]:** 1st conversion in injected sequence (when JL[1:0]=3, see note below)

Note: When JL[1:0]=3 (4 injected conversions in the sequencer), the ADC converts the channels in the following order: JSQ1[4:0], JSQ2[4:0], JSQ3[4:0], and JSQ4[4:0].

When JL=2 (3 injected conversions in the sequencer), the ADC converts the channels in the following order: JSQ2[4:0], JSQ3[4:0], and JSQ4[4:0].

When JL=1 (2 injected conversions in the sequencer), the ADC converts the channels in starting from JSQ3[4:0], and then JSQ4[4:0].

When JL=0 (1 injected conversion in the sequencer), the ADC converts only JSQ4[4:0] channel.

13.13.13 ADC injected data register x (ADC_JDRx) (x= 1..4)

Address offset: 0x3C - 0x48

Reset value: 0x0000 0000

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

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

Bits 15:0 **JDATA[15:0]:** Injected data

These bits are read-only. They contain the conversion result from injected channel x. The data are left -or right-aligned as shown in *Figure 48* and *Figure 49*.

13.13.14 ADC regular data register (ADC_DR)

Address offset: 0x4C

Reset value: 0x0000 0000

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

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

Bits 15:0 **DATA[15:0]:** Regular data

These bits are read-only. They contain the conversion result from the regular channels. The data are left- or right-aligned as shown in *Figure 48* and *Figure 49*.

13.13.15 ADC Common status register (ADC_CSR)

Address offset: 0x00 (this offset address is relative to ADC1 base address + 0x300)

Reset value: 0x0000 0000

This register provides an image of the status bits of the different ADCs. Nevertheless it is read-only and does not allow to clear the different status bits. Instead each status bit must be cleared by writing it to 0 in the corresponding ADC_SR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										OVR3	STRT3	JSTRT3	JEOC 3	EOC3	AWD3
										ADC3					
										r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		OVR2	STRT2	JSTRT 2	JEOC2	EOC2	AWD2	Reserved		OVR1	STRT1	JSTRT1	JEOC 1	EOC1	AWD1
		ADC2								ADC1					
		r	r	r	r	r	r			r	r	r	r	r	r

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

Bit 21 **OVR3:** Overrun flag of ADC3
This bit is a copy of the OVR bit in the ADC3_SR register.

Bit 20 **STRT3:** Regular channel Start flag of ADC3
This bit is a copy of the STRT bit in the ADC3_SR register.

Bit 19 **JSTRT3:** Injected channel Start flag of ADC3
This bit is a copy of the JSTRT bit in the ADC3_SR register.

Bit 18 **JEOC3:** Injected channel end of conversion of ADC3
This bit is a copy of the JEOC bit in the ADC3_SR register.

Bit 17 **EOC3:** End of conversion of ADC3
This bit is a copy of the EOC bit in the ADC3_SR register.

Bit 16 **AWD3:** Analog watchdog flag of ADC3
This bit is a copy of the AWD bit in the ADC3_SR register.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OVR2:** Overrun flag of ADC2
This bit is a copy of the OVR bit in the ADC2_SR register.

Bit 12 **STRT2:** Regular channel Start flag of ADC2
This bit is a copy of the STRT bit in the ADC2_SR register.

Bit 11 **JSTRT2:** Injected channel Start flag of ADC2
This bit is a copy of the JSTRT bit in the ADC2_SR register.

Bit 10 **JEOC2:** Injected channel end of conversion of ADC2
This bit is a copy of the JEOC bit in the ADC2_SR register.

Bit 9 **EOC2:** End of conversion of ADC2
This bit is a copy of the EOC bit in the ADC2_SR register.

Bit 8 **AWD2:** Analog watchdog flag of ADC2
This bit is a copy of the AWD bit in the ADC2_SR register.

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

Bit 5 **OVR1:** Overrun flag of ADC1
This bit is a copy of the OVR bit in the ADC1_SR register.

Bit 4 **STRT1:** Regular channel Start flag of ADC1
This bit is a copy of the STRT bit in the ADC1_SR register.

Bit 3 **JSTRT1:** Injected channel Start flag of ADC1
This bit is a copy of the JSTRT bit in the ADC1_SR register.

Bit 2 **JEOC1:** Injected channel end of conversion of ADC1
This bit is a copy of the JEOC bit in the ADC1_SR register.

Bit 1 **EOC1:** End of conversion of ADC1
This bit is a copy of the EOC bit in the ADC1_SR register.

Bit 0 **AWD1:** Analog watchdog flag of ADC1
This bit is a copy of the AWD bit in the ADC1_SR register.

13.13.16 ADC common control register (ADC_CCR)

Address offset: 0x04 (this offset address is relative to ADC1 base address + 0x300)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								TSVREFE	VBATE	Reserved				ADCPRE	
								rw	rw					rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMA[1:0]		DDS	Res.	DELAY[3:0]				Reserved			MULTI[4:0]				
rw	rw	rw		rw	rw	rw	rw				rw	rw	rw	rw	rw

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

Bit 23 **TSVREFE:** Temperature sensor and V_{REFINT} enable
This bit is set and cleared by software to enable/disable the temperature sensor and the V_{REFINT} channel.
0: Temperature sensor and V_{REFINT} channel disabled
1: Temperature sensor and V_{REFINT} channel enabled

Note: On STM32F42x and STM32F43x devices, VBATE must be disabled when TSVREFE is set. If both bits are set, only the VBAT conversion is performed.

Bit 22 **VBATE:** V_{BAT} enable
This bit is set and cleared by software to enable/disable the V_{BAT} channel.
0: V_{BAT} channel disabled
1: V_{BAT} channel enabled

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

Bits 17:16 **ADCPRE:** ADC prescaler

Set and cleared by software to select the frequency of the clock to the ADC. The clock is common for all the ADCs.

Note: 00: PCLK2 divided by 2
01: PCLK2 divided by 4
10: PCLK2 divided by 6
11: PCLK2 divided by 8

Bits 15:14 **DMA:** Direct memory access mode for multi ADC mode

This bit-field is set and cleared by software. Refer to the DMA controller section for more details.

00: DMA mode disabled
01: DMA mode 1 enabled (2 / 3 half-words one by one - 1 then 2 then 3)
10: DMA mode 2 enabled (2 / 3 half-words by pairs - 2&1 then 1&3 then 3&2)
11: DMA mode 3 enabled (2 / 3 bytes by pairs - 2&1 then 1&3 then 3&2)

Bit 13 **DDS:** DMA disable selection (for multi-ADC mode)

This bit is set and cleared by software.

0: No new DMA request is issued after the last transfer (as configured in the DMA controller). DMA bits are not cleared by hardware, however they must have been cleared and set to the wanted mode by software before new DMA requests can be generated.

1: DMA requests are issued as long as data are converted and DMA = 01, 10 or 11.

Bit 12 Reserved, must be kept at reset value.

Bit 11:8 **DELAY:** Delay between 2 sampling phases

Set and cleared by software. These bits are used in dual or triple interleaved modes.

0000: 5 * T_{ADCCLK}
0001: 6 * T_{ADCCLK}
0010: 7 * T_{ADCCLK}
...
1111: 20 * T_{ADCCLK}

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **MULTI[4:0]:** Multi ADC mode selection

These bits are written by software to select the operating mode.

– All the ADCs independent:
00000: Independent mode

– 00001 to 01001: Dual mode, ADC1 and ADC2 working together, ADC3 is independent
00001: Combined regular simultaneous + injected simultaneous mode
00010: Combined regular simultaneous + alternate trigger mode
00011: Reserved
00101: Injected simultaneous mode only
00110: Regular simultaneous mode only
00111: interleaved mode only
01001: Alternate trigger mode only

– 10001 to 11001: Triple mode: ADC1, 2 and 3 working together
10001: Combined regular simultaneous + injected simultaneous mode
10010: Combined regular simultaneous + alternate trigger mode
10011: Reserved
10101: Injected simultaneous mode only
10110: Regular simultaneous mode only
10111: interleaved mode only
11001: Alternate trigger mode only

All other combinations are reserved and must not be programmed

Note: In multi mode, a change of channel configuration generates an abort that can cause a loss of synchronization. It is recommended to disable the multi ADC mode before any configuration change.

13.13.17 ADC common regular data register for dual and triple modes (ADC_CDR)

Address offset: 0x08 (this offset address is relative to ADC1 base address + 0x300)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA2[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **DATA2[15:0]:** 2nd data item of a pair of regular conversions
- In dual mode, these bits contain the regular data of ADC2. Refer to *Dual ADC mode*.
- In triple mode, these bits contain alternatively the regular data of ADC2, ADC1 and ADC3. Refer to *Triple ADC mode*.

Bits 15:0 **DATA1[15:0]**: 1st data item of a pair of regular conversions
- In dual mode, these bits contain the regular data of ADC1. Refer to *Dual ADC mode*
- In triple mode, these bits contain alternatively the regular data of ADC1, ADC3 and ADC2. Refer to *Triple ADC mode*.

13.13.18 ADC register map

The following table summarizes the ADC registers.

Table 71. ADC global register map

Offset	Register
0x000 - 0x04C	ADC1
0x050 - 0x0FC	Reserved
0x100 - 0x14C	ADC2
0x118 - 0x1FC	Reserved
0x200 - 0x24C	ADC3
0x250 - 0x2FC	Reserved
0x300 - 0x308	Common registers

Table 72. ADC register map and reset values for each ADC

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**ADC_SR**	Reserved																										OVR	STRT	JSTRT	JEOC	EOC	AWD
	Reset value																											0	0	0	0	0	0
0x04	**ADC_CR1**	Reserved					OVRIE	RES[1:0]		AWDEN	JAWDEN	Reserved						DISC NUM [2:0]			JDISCEN	DISCEN	JAUTO	AWD SGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[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
0x08	**ADC_CR2**	Reserved	SWSTART	EXTEN[1:0]		EXTSEL [3:0]				Reserved	JSWSTART	JEXTEN[1:0]		JEXTSEL [3:0]				Reserved				ALIGN	EOCS	DDS	DMA	Reserved						CONT	ADON
	Reset value		0	0	0	0	0	0	0		0	0	0	0	0	0	0					0	0		0							0	0
0x0C	**ADC_SMPR1**	Sample time bits SMPx_x																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**ADC_SMPR2**	Sample time bits SMPx_x																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**ADC_JOFR1**	Reserved																				JOFFSET1[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x18	**ADC_JOFR2**	Reserved																				JOFFSET2[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**ADC_JOFR3**	Reserved																				JOFFSET3[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x20	**ADC_JOFR4**	Reserved																				JOFFSET4[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x24	**ADC_HTR**	Reserved																				HT[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x28	**ADC_LTR**	Reserved																				LT[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**ADC_SQR1**	Reserved								L[3:0]				Regular channel sequence SQx_x bits																			
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	**ADC_SQR2**	Reserved	Reserved	Regular channel sequence SQx_x bits																													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	**ADC_SQR3**	Reserved	Regular channel sequence SQx_x bits																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**ADC_JSQR**	Reserved										JL[1:0]		Injected channel sequence JSQx_x bits																			
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**ADC_JDR1**	Reserved																JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**ADC_JDR2**	Reserved																JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**ADC_JDR3**	Reserved																JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**ADC_JDR4**	Reserved																JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	**ADC_DR**	Reserved																Regular DATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 73. ADC register map and reset values (common ADC registers)

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**	Reserved										OVR	STRT	JSTRT	JEOC	EOC	AWD	Reserved		OVR	STRT	JSTRT	JEOC	EOC	AWD	Reserved		OVR	STRT	JSTRT	JEOC	EOC	AWD
	Reset value											0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
												ADC3								ADC2								ADC1					
0x04	**ADC_CCR**	Reserved								TSVREFE	VBATE	Reserved				ADCPRE[1:0]		DMA[1:0]		DDS	Reserved	DELAY [3:0]				Reserved		MULTI [4:0]					
	Reset value									0	0					0	0	0	0	0		0	0	0	0			0	0	0	0	0	0
0x08	**ADC_CDR**	Regular DATA2[15:0]																Regular DATA1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory map* for the register boundary addresses.

14 Digital-to-analog converter (DAC)

This section applies to the whole STM32F4xx family, unless otherwise specified.

14.1 DAC introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data could be left- or right-aligned. The DAC has two output channels, each with its own converter. In dual DAC channel mode, conversions could be done independently or simultaneously when both channels are grouped together for synchronous update operations. An input reference pin, V_{REF+} (shared with ADC) is available for better resolution.

14.2 DAC main features

- Two DAC converters: one output channel each
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave generation
- Triangular-wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel
- DMA underrun error detection
- External triggers for conversion
- Input voltage reference, V_{REF+}

Figure 64 shows the block diagram of a DAC channel and *Table 74* gives the pin description.

Figure 64. DAC channel block diagram

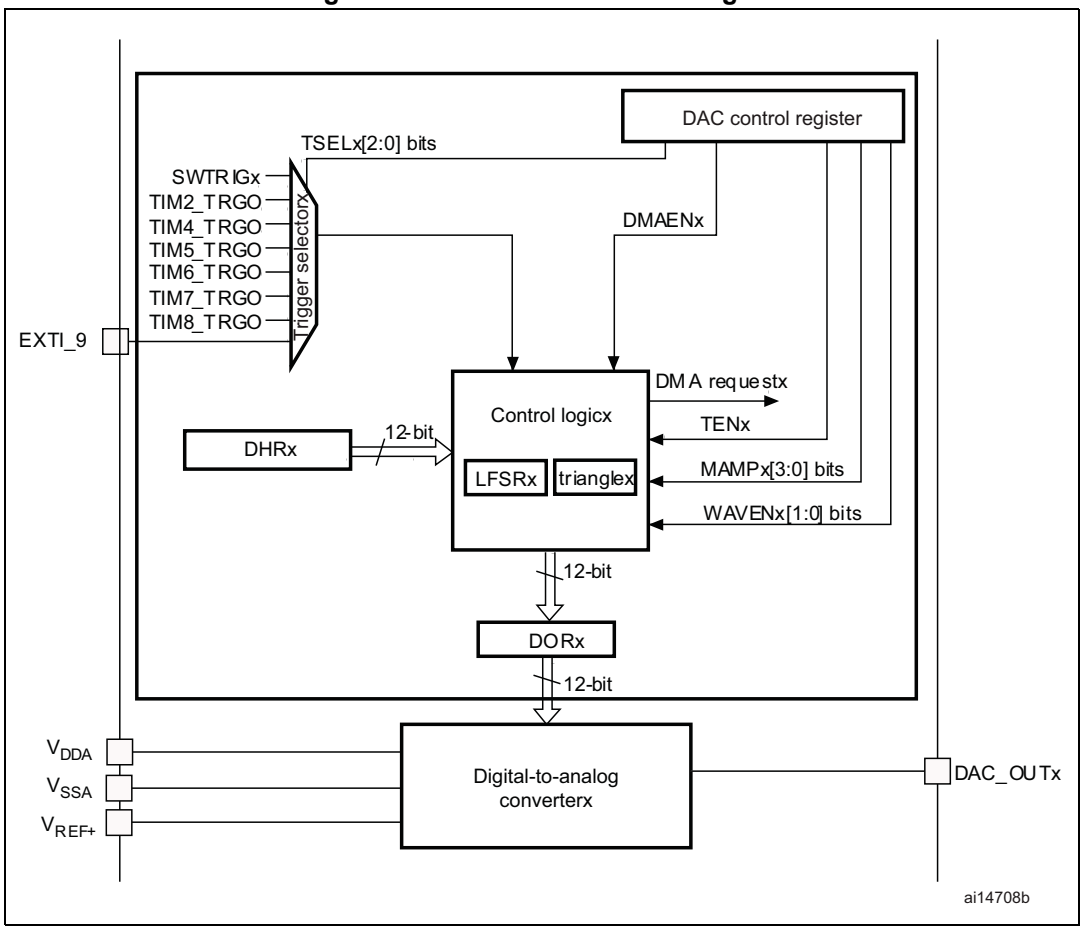


Table 74. DAC pins

Name	Signal type	Remarks
V_{REF+}	Input, analog reference positive	The higher/positive reference voltage for the DAC, 1.8 V $\leq V_{REF+} \leq V_{DDA}$
V_{DDA}	Input, analog supply	Analog power supply
V_{SSA}	Input, analog supply ground	Ground for analog power supply
DAC_OUTx	Analog output signal	DAC channelx analog output

Note: Once the DAC channelx is enabled, the corresponding GPIO pin (PA4 or PA5) is automatically connected to the analog converter output (DAC_OUTx). In order to avoid parasitic consumption, the PA4 or PA5 pin should first be configured to analog (AIN).

14.3 DAC functional description

14.3.1 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a startup time t_{WAKEUP}.

Note: The ENx bit enables the analog DAC Channelx macrocell only. The DAC Channelx digital interface is enabled even if the ENx bit is reset.

14.3.2 DAC output buffer enable

The DAC integrates two output buffers that can be used to reduce the output impedance, and to drive external loads directly without having to add an external operational amplifier. Each DAC channel output buffer can be enabled and disabled using the corresponding BOFFx bit in the DAC_CR register.

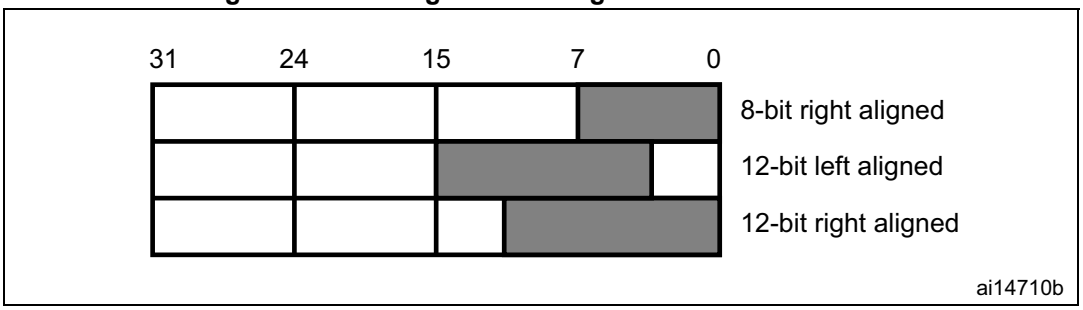
14.3.3 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channelx, there are three possibilities:
  - 8-bit right alignment: the software has to load data into the DAC_DHR8Rx [7:0] bits (stored into the DHRx[11:4] bits)
  - 12-bit left alignment: the software has to load data into the DAC_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)
  - 12-bit right alignment: the software has to load data into the DAC_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

Depending on the loaded DAC_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.

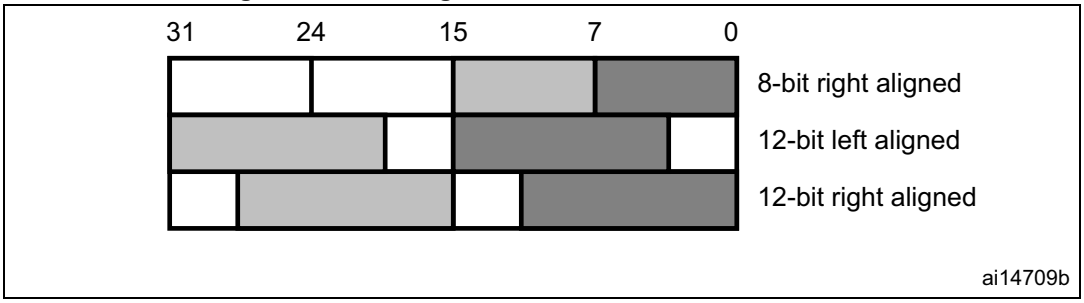
Figure 65. Data registers in single DAC channel mode



- Dual DAC channels, there are three possibilities:
  - 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)
  - 12-bit left alignment: data for DAC channel1 to be loaded into the DAC_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)
  - 12-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [27:16] bits (stored into the DHR2[11:0] bits)

Depending on the loaded DAC_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 DOR1 and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

Figure 66. Data registers in dual DAC channel mode



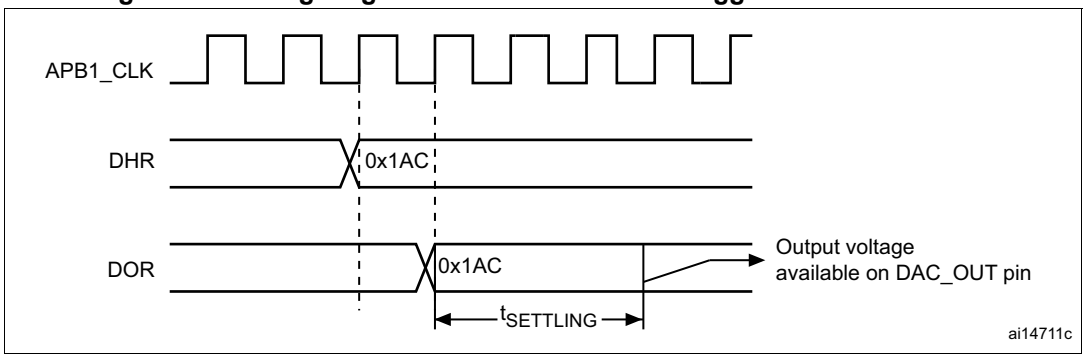
14.3.4 DAC conversion

The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12LD or DAC_DHR12LD).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one APB1 clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three APB1 clock cycles later.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

Figure 67. Timing diagram for conversion with trigger disabled TEN = 0



14.3.5 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{REF+}.

The analog output voltages on each DAC channel pin are determined by the following equation:

$$DACoutput = V_{REF} \times \frac{DOR}{4096}$$

14.3.6 DAC trigger selection

If the TENx control bit is set, conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[2:0] control bits determine which out of 8 possible events will trigger conversion as shown in *Table 75*.

Table 75. External triggers

Source	Type	TSEL[2:0]
Timer 6 TRGO event	Internal signal from on-chip timers	000
Timer 8 TRGO event		001
Timer 7 TRGO event		010
Timer 5 TRGO event		011
Timer 2 TRGO event		100
Timer 4 TRGO event		101
EXTI line9	External pin	110
SWTRIG	Software control bit	111

Each time a DAC interface detects a rising edge on the selected timer TRGO output, or on the selected external interrupt line 9, the last data stored into the DAC_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three APB1 cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

Note: 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.

14.3.7 DMA request

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

A DAC DMA request is generated when an external trigger (but not a software trigger) occurs while the DMAENx bit is set. The value of the DAC_DHRx register is then transferred into the DAC_DORx register.
In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, the user should set only the corresponding DMAENx bit. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgement for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. DMA data transfers are then disabled and no further DMA request is treated. The DAC channelx continues to convert old data.

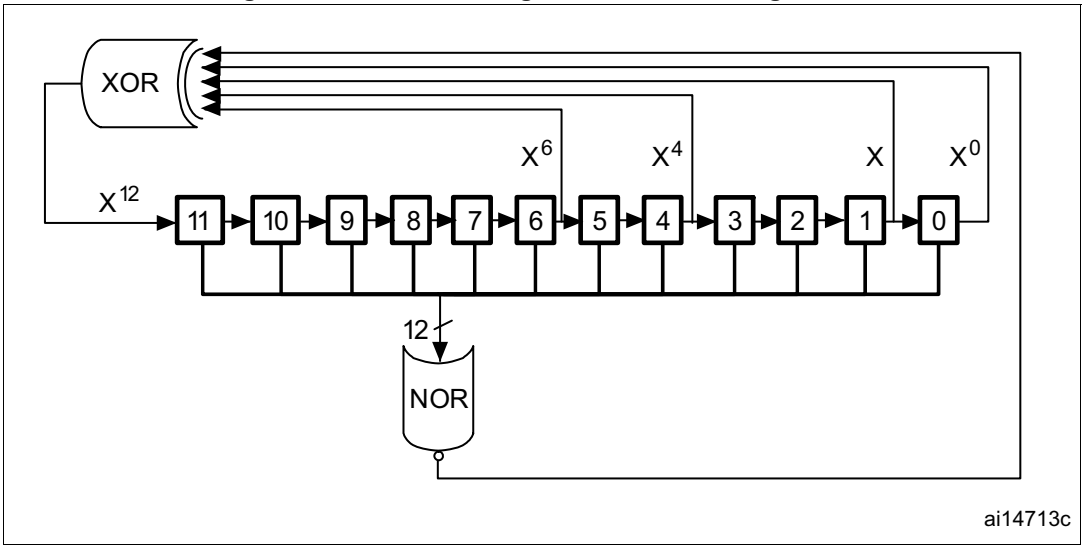
The software should clear the DMAUDRx flag by writing “1”, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software should modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion could be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channelx, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC_CR register is enabled.

14.3.8 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to “01”. The preloaded value in LFSR is 0xAAA. This register is updated three APB1 clock cycles after each trigger event, following a specific calculation algorithm.

Figure 68. DAC LFSR register calculation algorithm

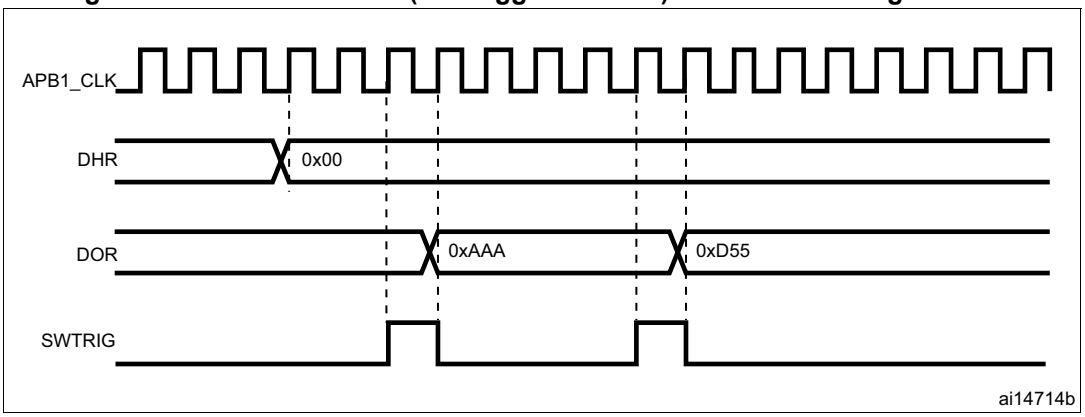


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then stored into the DAC_DORx register.

If LFSR is 0x0000, a ‘1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Figure 69. 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.

14.3.9 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to “10”. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three APB1 clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is stored into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

Figure 70. DAC triangle wave generation

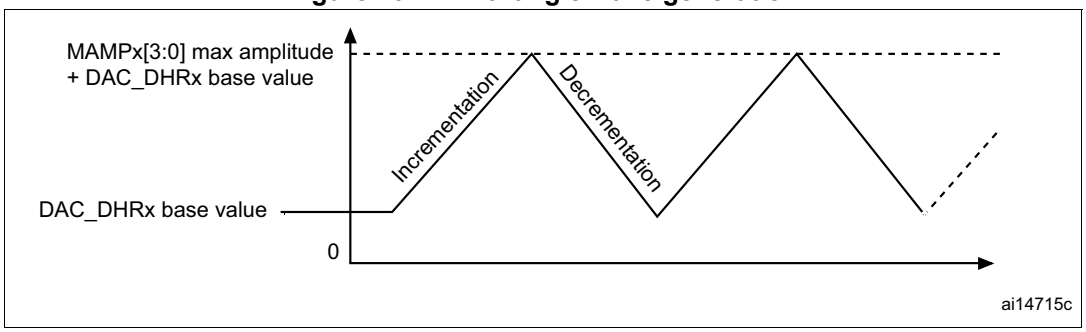
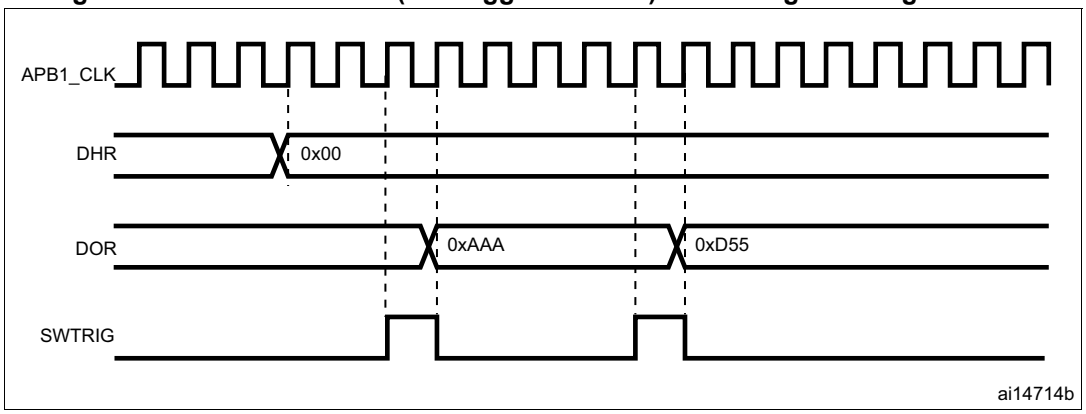


Figure 71. DAC conversion (SW trigger enabled) with triangle wave generation



Note: The DAC trigger must be enabled for noise 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.

14.4 Dual DAC channel conversion

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time.

Eleven possible conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DHRx registers if needed.

All modes are described in the paragraphs below.

14.4.1 Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three APB1 clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three APB1 clock cycles later).

14.4.2 Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “01” and the same LFSR mask value in the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DHR12RD, DHR12LD or DHR8RD)

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). Then the LFSR2 counter is updated.

14.4.3 Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “01” and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). Then the LFSR2 counter is updated.

14.4.4 Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “1x” and the same maximum amplitude value in the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

14.4.5 Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “1x” and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

14.4.6 Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

In this configuration, one APB1 clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

14.4.7 Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three APB1 clock cycles).

14.4.8 Simultaneous trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “01” and the same LFSR mask value in the MAMPx[3:0] bits
- Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD)

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The LFSR2 counter is then updated.

14.4.9 Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “01” and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The LFSR1 counter is then updated.
At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The LFSR2 counter is then updated.

14.4.10 Simultaneous trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “1x” and the same maximum amplitude value using the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The DAC channel1 triangle counter is then updated.
At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

14.4.11 Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel 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 APB1 clock cycles later). Then the DAC channel1 triangle counter is updated.
At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). Then the DAC channel2 triangle counter is updated.

14.5 DAC registers

Refer to *Section 1.1: List of abbreviations for registers for registers* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

14.5.1 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
Reserved		DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[2:0]			TEN2	BOFF2	EN2
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29 **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable
This bit is set and cleared by software.
0: DAC channel2 DMA underrun interrupt disabled
1: DAC channel2 DMA underrun interrupt enabled

Bit 28 **DMAEN2**: DAC channel2 DMA enable
This bit is set and cleared by software.
0: DAC channel2 DMA mode disabled
1: DAC channel2 DMA mode enabled

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

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)

Bits 21:19 **TSEL2[2:0]**: DAC channel2 trigger selection

These bits select the external event used to trigger DAC channel2

000: Timer 6 TRGO event

001: Timer 8 TRGO event

010: Timer 7 TRGO event

011: Timer 5 TRGO event

100: Timer 2 TRGO event

101: Timer 4 TRGO event

110: External line9

111: Software trigger

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).

Bit 18 **TEN2**: DAC channel2 trigger enable

This bit is set and cleared by software to enable/disable DAC channel2 trigger

0: DAC channel2 trigger disabled and data written into the DAC_DHRx register are transferred one APB1 clock cycle later to the DAC_DOR2 register

1: DAC channel2 trigger enabled and data from the DAC_DHRx register are transferred three APB1 clock cycles later to the DAC_DOR2 register

Note: When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DOR2 register takes only one APB1 clock cycle.

Bit 17 **BOFF2**: DAC channel2 output buffer disable

This bit is set and cleared by software to enable/disable DAC channel2 output buffer.

0: DAC channel2 output buffer enabled

1: DAC channel2 output buffer disabled

Bit 16 **EN2**: DAC channel2 enable

This bit is set and cleared by software to enable/disable DAC channel2.

0: DAC channel2 disabled

1: DAC channel2 enabled

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.

0: DAC channel1 DMA Underrun Interrupt disabled

1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.

0: DAC channel1 DMA mode disabled

1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

These bits are set and cleared by software.

00: wave generation disabled
01: Noise wave generation enabled
1x: Triangle wave generation enabled

Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 5:3 **TSEL1[2:0]**: DAC channel1 trigger selection

These bits select the external event used to trigger DAC channel1.

000: Timer 6 TRGO event
001: Timer 8 TRGO event
010: Timer 7 TRGO event
011: Timer 5 TRGO event
100: Timer 2 TRGO event
101: Timer 4 TRGO event
110: External line9
111: Software trigger

Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 2 **TEN1**: DAC channel1 trigger enable

This bit is set and cleared by software to enable/disable DAC channel1 trigger.

0: DAC channel1 trigger disabled and data written into the DAC_DHRx register are transferred one APB1 clock cycle later to the DAC_DOR1 register
1: DAC channel1 trigger enabled and data from the DAC_DHRx register are transferred three APB1 clock cycles later to the DAC_DOR1 register

Note: When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DOR1 register takes only one APB1 clock cycle.

Bit 1 **BOFF1**: DAC channel1 output buffer disable

This bit is set and cleared by software to enable/disable DAC channel1 output buffer.

0: DAC channel1 output buffer enabled
1: DAC channel1 output buffer disabled

Bit 0 **EN1**: DAC channel1 enable

This bit is set and cleared by software to enable/disable DAC channel1.

0: DAC channel1 disabled
1: DAC channel1 enabled

14.5.2 DAC software trigger register (DAC_SWTRIGR)

Address offset: 0x04
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														SWTRIG2	SWTRIG1
														w	w

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SWTRIG2**: DAC channel2 software trigger

This bit is set and cleared by software to enable/disable the software trigger.
0: Software trigger disabled
1: Software trigger enabled

Note: This bit is cleared by hardware (one APB1 clock cycle later) once the DAC_DHR2 register value has been loaded into the DAC_DOR2 register.

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set and cleared by software to enable/disable the software trigger.
0: Software trigger disabled
1: Software trigger enabled

Note: This bit is cleared by hardware (one APB1 clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

14.5.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

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.

14.5.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

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

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

14.5.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel1.

14.5.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				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 which specifies 12-bit data for DAC channel2.

14.5.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Reserved			
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.

14.5.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)

Address offset: 0x1C

Reset value: 0x0000 0000

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

14.5.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 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.

14.5.10 DUAL DAC 12-bit left aligned data holding register (DAC_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

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

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

14.5.11 DUAL DAC 8-bit right aligned data holding register (DAC_DHR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								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.

14.5.12 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

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

Bit 11:0 **DACC1DOR[11:0]**: DAC channel1 data output
These bits are read-only, they contain data output for DAC channel1.

14.5.13 DAC channel2 data output register (DAC_DOR2)

Address offset: 0x30
Reset value: 0x0000 0000

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

14.5.14 DAC status register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DMAUDR2	Reserved												
		rc_w1													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		DMAUDR1	Reserved												
		rc_w1													

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

Bit 29 **DMAUDR2**: DAC channel2 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).
0: No DMA underrun error condition occurred for DAC channel2
1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate)

Bits 28:14 Reserved, must be kept at reset value.

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).
0: No DMA underrun error condition occurred for DAC channel1
1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bits 12:0 Reserved, must be kept at reset value.

14.5.15 DAC register map

Table 76 summarizes the DAC registers.

Table 76. DAC 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	DAC_CR	Reserved		DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE 2[2:0]		TSEL2[2:0]			TEN2	BOFF2	EN2	Reserved		DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE 1[2:0]		TSEL1[2 :0]			TEN1	BOFF1	EN1
0x04	DAC_ SWTRIGR	Reserved																														SWTRIG2	SWTRIG1
0x08	DAC_ DHR12R1	Reserved																				DACC1DHR[11:0]											
0x0C	DAC_ DHR12L1	Reserved																DACC1DHR[11:0]												Reserved			
0x10	DAC_ DHR8R1	Reserved																								DACC1DHR[7:0]							
0x14	DAC_ DHR12R2	Reserved																				DACC2DHR[11:0]											
0x18	DAC_ DHR12L2	Reserved																DACC2DHR[11:0]												Reserved			

Table 76. DAC 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
0x1C	DAC_DHR8R2	Reserved																								DACC2DHR[7:0]							
0x20	DAC_DHR12RD	Reserved				DACC2DHR[11:0]												Reserved				DACC1DHR[11:0]											
0x24	DAC_DHR12LD	DACC2DHR[11:0]												Reserved				DACC1DHR[11:0]								Reserved							
0x28	DAC_DHR8RD	Reserved																DACC2DHR[7:0]								DACC1DHR[7:0]							
0x2C	DAC_DOR1	Reserved																				DACC1DOR[11:0]											
0x30	DAC_DOR2	Reserved																				DACC2DOR[11:0]											
0x34	DAC_SR	Reserved		DMAUDR2	Reserved															DMAUDR1	Reserved												

Refer to *Section 2.3: Memory map* for the register boundary addresses.

15 Digital camera interface (DCMI)

This section applies to all STM32F4xx devices, unless otherwise specified.

15.1 DCMI introduction

The digital camera is a synchronous parallel interface able to receive a high-speed data flow from an external 8-, 10-, 12- or 14-bit CMOS camera module. It supports different data formats: YCbCr4:2:2/RGB565 progressive video and compressed data (JPEG).

This interface is for use with black & white cameras, X24 and X5 cameras, and it is assumed that all pre-processing like resizing is performed in the camera module.

15.2 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

15.3 DCMI pins

Table 77 shows the DCMI pins.

Table 77. DCMI pins

Name	Signal type
D[0:13]	Data inputs
HSYNC	Horizontal synchronization input
VSYNC	Vertical synchronization input
PIXCLK	Pixel clock input

15.4 DCMI clocks

The digital camera interface uses two clock domains PIXCLK and HCLK. The signals generated with PIXCLK are sampled on the rising edge of HCLK once they are stable. An enable signal is generated in the HCLK domain, to indicate that data coming from the camera are stable and can be sampled. The minimum PIXCLK period must be higher than 2.5 HCLK periods.

15.5 DCMI functional overview

The digital camera interface is a synchronous parallel interface that can receive high-speed (up to 54 Mbytes/s) data flows. It consists of up to 14 data lines (D13-D0) and a pixel clock line (PIXCLK). The pixel clock has a programmable polarity, so that data can be captured on either the rising or the falling edge of the pixel clock.

The data are packed into a 32-bit data register (DCMI_DR) and then transferred through a general-purpose DMA channel. The image buffer is managed by the DMA, not by the camera interface.

The data received from the camera can be organized in lines/frames (raw YUB/RGB/Bayer modes) or can be a sequence of JPEG images. To enable JPEG image reception, the JPEG bit (bit 3 of DCMI_CR register) must be set.

The data flow is synchronized either by hardware using the optional HSYNC (horizontal synchronization) and VSYNC (vertical synchronization) signals or by synchronization codes embedded in the data flow.

Figure 72 shows the DCMI block diagram.

Figure 72. DCMI block diagram

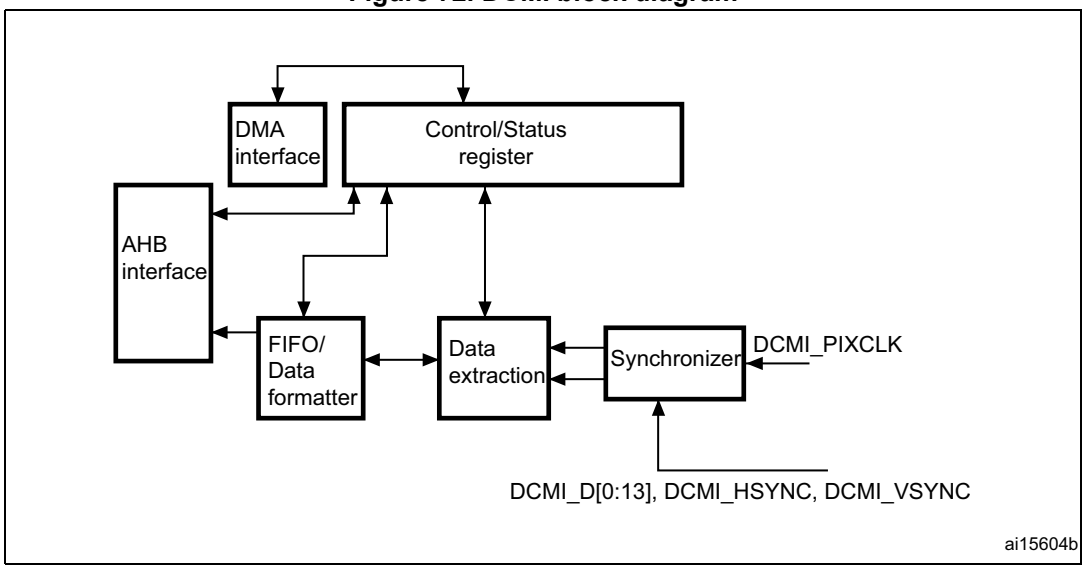
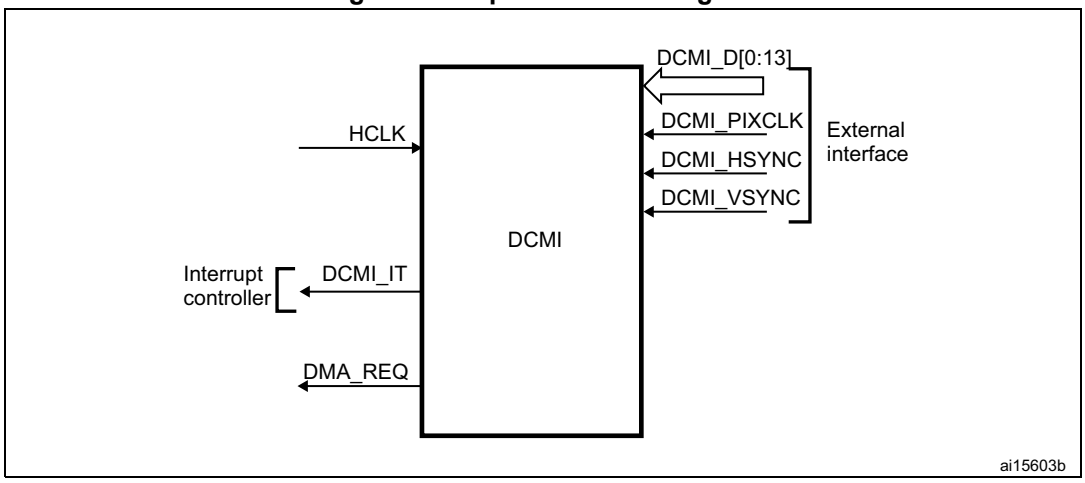


Figure 73. Top-level block diagram



15.5.1 DMA interface

The DMA interface is active when the CAPTURE bit in the DCMI_CR register is set. A DMA request is generated each time the camera interface receives a complete 32-bit data block in its register.

15.5.2 DCMI physical interface

The interface is composed of 11/13/15/17 inputs. Only the Slave mode is supported.

The camera interface can capture 8-bit, 10-bit, 12-bit or 14-bit data depending on the EDM[1:0] bits in the DCMI_CR register. If less than 14 bits are used, the unused data pins must not be assigned to DCMI interface through GPIO alternate functions.

Table 78. DCMI signals

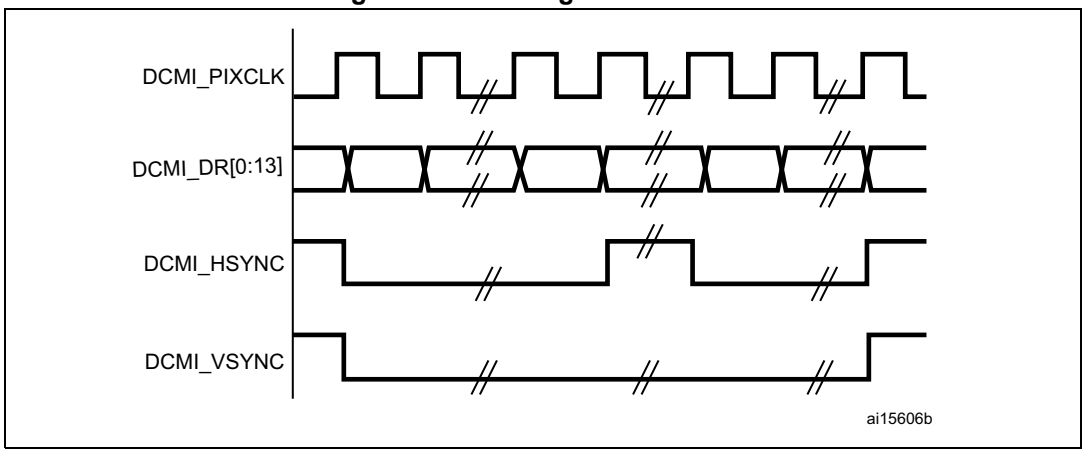
Signal name		Signal description
8 bits 10 bits 12 bits 14 bits	D[0..7] D[0..9] D[0..11] D[0..13]	Data
PIXCLK		Pixel clock
HSYNC		Horizontal synchronization / Data valid
VSYNC		Vertical synchronization

The data are synchronous with PIXCLK and change on the rising/falling edge of the pixel clock depending on the polarity.

The HSYNC signal indicates the start/end of a line.

The VSYNC signal indicates the start/end of a frame

Figure 74. DCMI signal waveforms



1. The capture edge of DCMI_PIXCLK is the falling edge, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.

1. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

8-bit data

When EDM[1:0] in DCMI_CR are programmed to “00” the interface captures 8 LSB’s at its input (D[0:7]) and stores them as 8-bit data. The D[13:8] inputs are ignored. In this case, to capture a 32-bit word, the camera interface takes four pixel clock cycles.

The first captured data byte is placed in the LSB position in the 32-bit word and the 4th captured data byte is placed in the MSB position in the 32-bit word. *Table 79* gives an example of the positioning of captured data bytes in two 32-bit words.

Table 79. Positioning of captured data bytes in 32-bit words (8-bit width)

Byte address	31:24	23:16	15:8	7:0
0	D_{n+3}[7:0]	D_{n+2}[7:0]	D_{n+1}[7:0]	D_n[7:0]
4	D_{n+7}[7:0]	D_{n+6}[7:0]	D_{n+5}[7:0]	D_{n+4}[7:0]

10-bit data

When EDM[1:0] in DCMI_CR are programmed to “01”, the camera interface captures 10-bit data at its input D[0..9] and stores them as the 10 least significant bits of a 16-bit word. The remaining most significant bits in the DCMI_DR register (bits 11 to 15) are cleared to zero. So, in this case, a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in *Table 80*.

Table 80. Positioning of captured data bytes in 32-bit words (10-bit width)

Byte address	31:26	25:16	15:10	9:0
0	0	D_{n+1}[9:0]	0	D_n[9:0]
4	0	D_{n+3}[9:0]	0	D_{n+2}[9:0]

12-bit data

When EDM[1:0] in DCMI_CR are programmed to “10”, the camera interface captures the 12-bit data at its input D[0..11] and stores them as the 12 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in *Table 81*.

Table 81. Positioning of captured data bytes in 32-bit words (12-bit width)

Byte address	31:28	27:16	15:12	11:0
0	0	D_{n+1}[11:0]	0	D_n[11:0]
4	0	D_{n+3}[11:0]	0	D_{n+2}[11:0]

14-bit data

When EDM[1:0] in DCMI_CR are programmed to “11”, the camera interface captures the 14-bit data at its input D[0..13] and stores them as the 14 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in *Table 82*.

Table 82. Positioning of captured data bytes in 32-bit words (14-bit width)

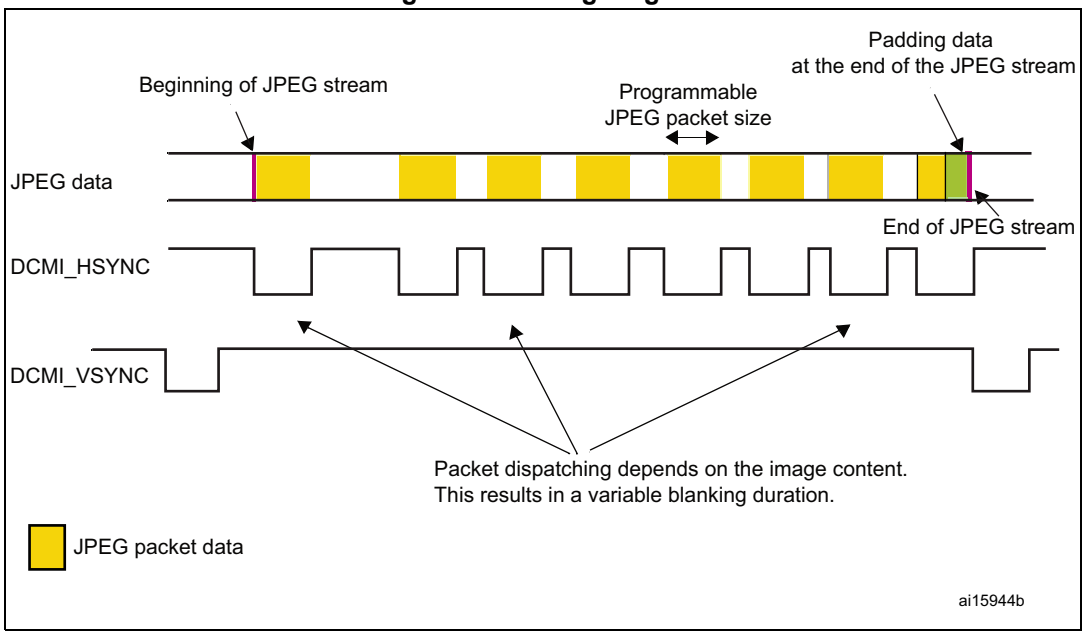
Byte address	31:30	29:16	15:14	13:0
0	0	D_{n+1}[13:0]	0	D_n[13:0]
4	0	D_{n+3}[13:0]	0	D_{n+2}[13:0]

15.5.3 Synchronization

The digital camera interface supports embedded or hardware (HSYNC & VSYNC) synchronization. When embedded synchronization is used, it is up to the digital camera module to make sure that the 0x00 and 0xFF values are used ONLY for synchronization (not in data). Embedded synchronization codes are supported only for the 8-bit parallel data interface width (that is, in the DCMI_CR register, the EDM[1:0] bits should be cleared to “00”).

For compressed data, the DCMI supports only the hardware synchronization mode. In this case, VSYNC is used as a start/end of the image, and HSYNC is used as a Data Valid signal. *Figure 75* shows the corresponding timing diagram.

Figure 75. Timing diagram



Hardware synchronization mode

In hardware synchronisation mode, the two synchronization signals (HSYNC/VSYNC) are used.

Depending on the camera module/mode, data may be transmitted during horizontal/vertical synchronisation periods. The HSYNC/VSYNC signals act like blanking signals since all the data received during HSYNC/VSYNC active periods are ignored.

In order to correctly transfer images into the DMA/RAM buffer, data transfer is synchronized with the VSYNC signal. When the hardware synchronisation mode is selected, and capture is enabled (CAPTURE bit set in DCMI_CR), data transfer is synchronized with the deactivation of the VSYNC signal (next start of frame).

Transfer can then be continuous, with successive frames transferred by DMA to successive buffers or the same/circular buffer. To allow the DMA management of successive frames, a VSIF (Vertical synchronization interrupt flag) is activated at the end of each frame.

Embedded data synchronization mode

In this synchronisation mode, the data flow is synchronised using 32-bit codes embedded in the data flow. These codes use the 0x00/0xFF values that are *not* used in data anymore. There are 4 types of codes, all with a 0xFF0000XY format. The embedded synchronization codes are supported only in 8-bit parallel data width capture (in the DCMI_CR register, the EDM[1:0] bits should be programmed to “00”). For other data widths, this mode generates unpredictable results and must not be used.

Note: Camera modules can have 8 such codes (in interleaved mode). For this reason, the interleaved mode is not supported by the camera interface (otherwise, every other half-frame would be discarded).

- Mode 2

  Four embedded codes signal the following events
  - Frame start (FS)
  - Frame end (FE)
  - Line start (LS)
  - Line end (LE)

  The XY values in the 0xFF0000XY format of the four codes are programmable (see *Section 15.8.7: DCMI embedded synchronization code register (DCMI_ESCR)*).

  A 0xFF value programmed as a “frame end” means that all the unused codes are interpreted as valid frame end codes.

  In this mode, once the camera interface has been enabled, the frame capture starts after the first occurrence of the frame end (FE) code followed by a frame start (FS) code.
- Mode 1

  An alternative coding is the camera mode 1. This mode is ITU656 compatible.

  The codes signal another set of events:
  - SAV (active line) - line start
  - EAV (active line) - line end
  - SAV (blanking) - end of line during interframe blanking period
  - EAV (blanking) - end of line during interframe blanking period

This mode can be supported by programming the following codes:

- FS ≤ 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.

15.5.4 Capture modes

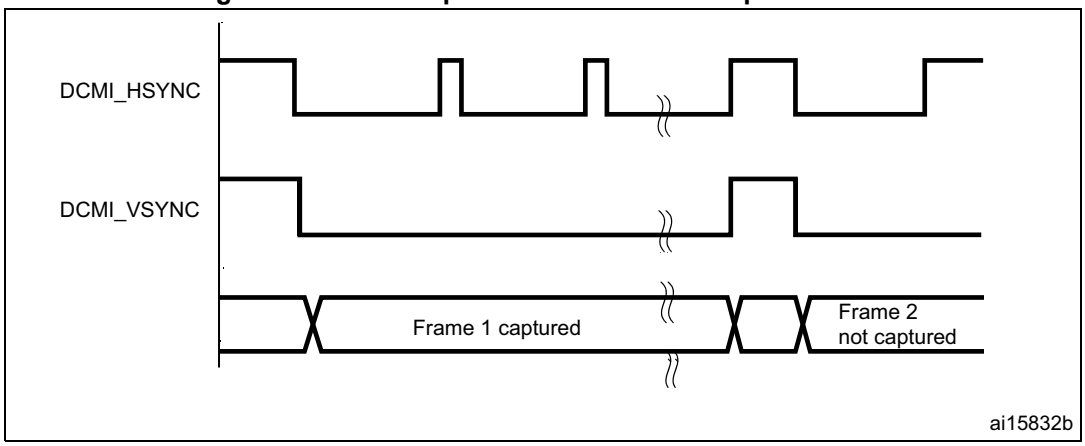
This interface supports two types of capture: snapshot (single frame) and continuous grab.

Snapshot mode (single frame)

In this mode, a single frame is captured (CM = '1' in the DCMI_CR register). After the CAPTURE bit is set in DCMI_CR, the interface waits for the detection of a start of frame before sampling the data. The camera interface is automatically disabled (CAPTURE bit cleared in DCMI_CR) after receiving the first complete frame. An interrupt is generated (IT_FRAME) if it is enabled.

In case of an overrun, the frame is lost and the CAPTURE bit is cleared.

Figure 76. Frame capture waveforms in Snapshot mode

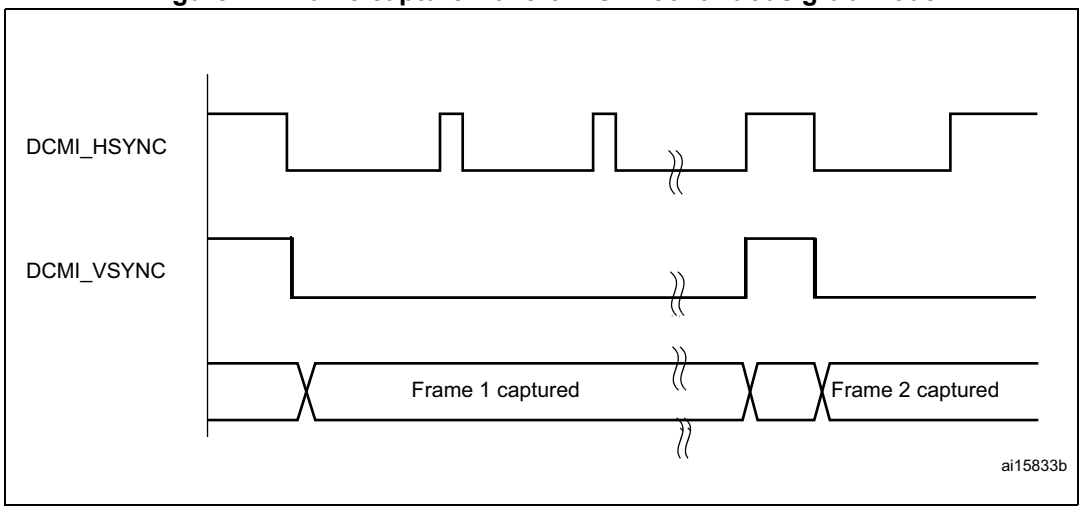


1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

Continuous grab mode

In this mode (CM bit = ‘0’ in DCMI_CR), once the CAPTURE bit has been set in DCMI_CR, the grabbing process starts on the next VSYNC or embedded frame start depending on the mode. The process continues until the CAPTURE bit is cleared in DCMI_CR. Once the CAPTURE bit has been cleared, the grabbing process continues until the end of the current frame.

Figure 77. 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.

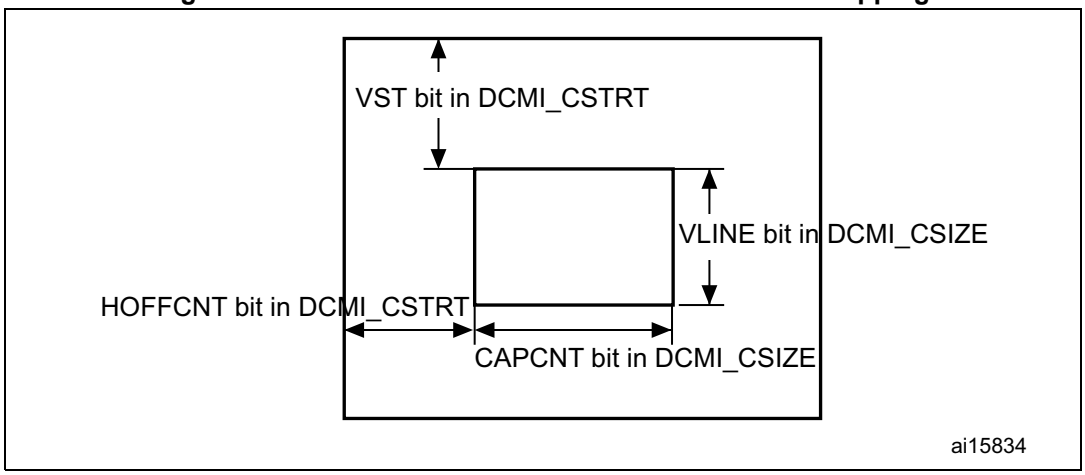
In continuous grab mode, you can configure the FCRC bits in DCMI_CR to grab all pictures, every second picture or one out of four pictures to decrease the frame capture rate.

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.

15.5.5 Crop feature

With the crop feature, the camera interface can select a rectangular window from the received image. The start (upper left corner) coordinates and size (horizontal dimension in number of pixel clocks and vertical dimension in number of lines) are specified using two 32-bit registers (DCMI_CWSTRT and DCMI_CWSIZE). The size of the window is specified in number of pixel clocks (horizontal dimension) and in number of lines (vertical dimension).

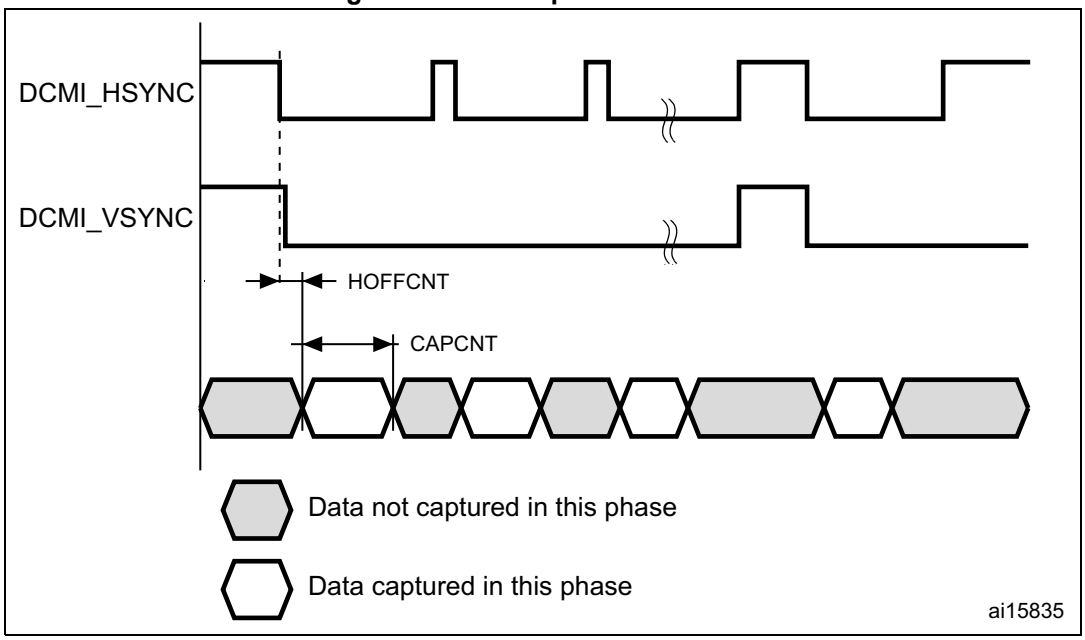
Figure 78. Coordinates and size of the window after cropping



These registers specify the coordinates of the starting point of the capture window as a line number (in the frame, starting from 0) and a number of pixel clocks (on the line, starting from 0), and the size of the window as a line number and a number of pixel clocks. The CAPCNT value can only be a multiple of 4 (two least significant bits are forced to 0) to allow the correct transfer of data through the DMA.

If the VSYNC signal goes active before the number of lines is specified in the DCMI_CWSIZE register, then the capture stops and an IT_FRAME interrupt is generated when enabled.

Figure 79. 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.

15.5.6 JPEG format

To allow JPEG image reception, it is necessary to set the JPEG bit in the DCMI_CR register. JPEG images are not stored as lines and frames, so the VSYNC signal is used to start the capture while HSYNC serves as a data enable signal. The number of bytes in a line may not be a multiple of 4, you should therefore be careful when handling this case since a DMA request is generated each time a complete 32-bit word has been constructed from the captured data. When an end of frame is detected and the 32-bit word to be transferred has not been completely received, the remaining data are padded with '0s' and a DMA request is generated.

The crop feature and embedded synchronization codes cannot be used in the JPEG format.

15.5.7 FIFO

A four-word FIFO is implemented to manage data rate transfers on the AHB. The DCMI features a simple FIFO controller with a read pointer incremented each time the camera interface reads from the AHB, and a write pointer incremented each time the camera interface writes to the FIFO. There is no overrun protection to prevent the data from being overwritten if the AHB interface does not sustain the data transfer rate.

In case of overrun or errors in the synchronization signals, the FIFO is reset and the DCMI interface waits for a new start of frame.

15.6 Data format description

15.6.1 Data formats

Three types of data are supported:

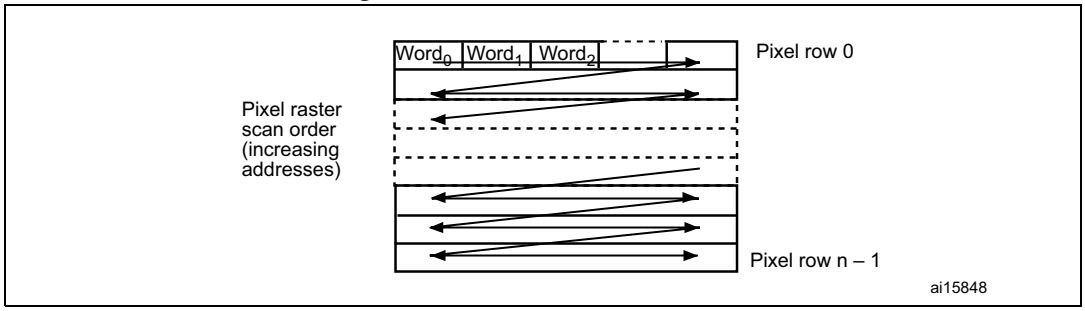
- 8-bit progressive video: either monochrome or raw Bayer format
- YCbCr 4:2:2 progressive video
- RGB565 progressive video. A pixel coded in 16 bits (5 bits for blue, 5 bits for red, 6 bits for green) takes two clock cycles to be transferred.

Compressed data: JPEG

For B&W, YCbCr or RGB data, the maximum input size is 2048 × 2048 pixels. No limit in JPEG compressed mode.

For monochrome, RGB & YCbCr, the frame buffer is stored in raster mode. 32-bit words are used. Only the little endian format is supported.

Figure 80. Pixel raster scan order



15.6.2 Monochrome format

Characteristics:

- Raster format
- 8 bits per pixel

Table 83 shows how the data are stored.

Table 83. 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

15.6.3 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 84 shows how the data are stored.

Table 84. 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

15.6.4 YCbCr format

Characteristics:

- Raster format
- YCbCr 4:2:2
- Interleaved: one Buffer: Y, Cb & Cr interleaved: CbYCrYCbYCr, etc.

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). Each component is encoded in 8 bits. Luma and chroma are stored together (interleaved) as shown in *Table 85*.

Table 85. 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

15.7 DCMI interrupts

Five interrupts are generated. All interrupts are maskable by software. The global interrupt (IT_DCMI) is the OR of all the individual interrupts. *Table 86* gives the list of all interrupts.

Table 86. 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

15.8 DCMI register description

All DCMI registers have to be accessed as 32-bit words, otherwise a bus error occurs.

15.8.1 DCMI control register 1 (DCMI_CR)

Address offset: 0x00

Reset value: 0x0000 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	ENABLE	Reserved		EDM		FCRC		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
																	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 14 **ENABLE:** DCMI enable
0: DCMI disabled
1: DCMI enabled
Note: The DCMI configuration registers should be programmed correctly before enabling this Bit

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

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 VSYNC pin when the data are not valid on the parallel interface.
0: VSYNC active low
1: VSYNC active high

Bit 6 **HSPOL:** Horizontal synchronization polarity
This bit indicates the level on the HSYNC pin when the data are not valid on the parallel interface.
0: HSYNC active low
1: HSYNC active high

Bit 5 **PCKPOL:** Pixel clock polarity
This bit configures the capture edge of the pixel clock
0: Falling edge active.
1: Rising edge active.

Bit 4 **ESS:** Embedded synchronization select
0: Hardware synchronization data capture (frame/line start/stop) is synchronized with the HSYNC/VSYNC signals.
1: Embedded synchronization data capture is synchronized with synchronization codes embedded in the data flow.
Note: Valid only for 8-bit parallel data. HSPOL/VSPOL are ignored when the ESS bit is set.
This bit is disabled in JPEG mode.

Bit 3 **JPEG:** JPEG format
0: Uncompressed video format
1: This bit is used for JPEG data transfers. The HSYNC signal is used as data enable. The crop and embedded synchronization features (ESS bit) cannot be used in this mode.

Bits 2 **CROP:** Crop feature
0: The full image is captured. In this case the total number of bytes in an image frame should be a multiple of 4
1: Only the data inside the window specified by the crop register will be captured. If the size of the crop window exceeds the picture size, then only the picture size is captured.

Bit 1 **CM:** Capture mode

0: Continuous grab mode - The received data are transferred into the destination memory through the DMA. The buffer location and mode (linear or circular buffer) is controlled through the system DMA.

1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame and then transfers a single frame through the DMA. At the end of the frame, the CAPTURE bit is automatically reset.

Bit 0 **CAPTURE:** Capture enable

0: Capture disabled.

1: Capture enabled.

The camera interface waits for the first start of frame, then a DMA request is generated to transfer the received data into the destination memory.

In snapshot mode, the CAPTURE bit is automatically cleared at the end of the 1st frame received.

In continuous grab mode, if the software clears this bit while a capture is ongoing, the bit will be effectively cleared after the frame end.

Note: The DMA controller and all DCMI configuration registers should be programmed correctly before enabling this bit.

15.8.2 DCMI status register (DCMI_SR)

Address offset: 0x04

Reset value: 0x0000 0x0000

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

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 VSYNC pin with the correct programmed polarity. When embedded synchronization codes are used, the meaning of this bit is the following:
0: active frame
1: synchronization between frames
In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

Bit 0 **HSYNC**
This bit gives the state of the HSYNC pin with the correct programmed polarity. When embedded synchronization codes are used, the meaning of this bit is the following:
0: active line
1: synchronization between lines
In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

15.8.3 DCMI raw interrupt status register (DCMI_RIS)

Address offset: 0x08

Reset value: 0x0000 0x0000

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

DCMI_RIS gives the raw interrupt status and is accessible in read only. When read, this register returns the status of the corresponding interrupt before masking with the DCMI_IER register value.

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_RIS:** Line raw interrupt status

This bit gets set when the HSYNC signal changes from the inactive state to the active state. It goes high even if the line is not valid.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit in DCMI_CR is set.

It is cleared by writing a ‘1’ to the LINE_ISC bit in DCMI_ICR.

Bit 3 **VSYNC_RIS:** VSYNC raw interrupt status

This bit is set when the VSYNC signal changes from the inactive state to the active state.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMI_CR.

It is cleared by writing a ‘1’ to the VSYNC_ISC bit in DCMI_ICR.

Bit 2 **ERR_RIS:** Synchronization error raw interrupt status

0: No synchronization error detected

1: Embedded synchronization characters are not received in the correct order.

This bit is valid only in the embedded synchronization mode. It is cleared by writing a ‘1’ to the ERR_ISC bit in DCMI_ICR.

Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_RIS:** Overrun raw interrupt status

0: No data buffer overrun occurred

1: A data buffer overrun occurred and the data FIFO is corrupted.

This bit is cleared by writing a ‘1’ to the OVR_ISC bit in DCMI_ICR.

Bit 0 **FRAME_RIS:** Capture complete raw interrupt status

0: No new capture

1: A frame has been captured.

This bit is set when a frame or window has been captured.

In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (e.g. window cropped outside the frame).

This bit is cleared by writing a ‘1’ to the FRAME_ISC bit in DCMI_ICR.

15.8.4 DCMI interrupt enable register (DCMI_IER)

Address offset: 0x0C

Reset value: 0x0000 0x0000

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

The DCMI_IER register is used to enable interrupts. When one of the DCMI_IER bits is set, the corresponding interrupt is enabled. This register is accessible in both read and write.

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:** VSYNC interrupt enable
0: No interrupt generation
1: An interrupt is generated on each VSYNC transition from the inactive to the active state
The active state of the VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR_IE:** Synchronization error interrupt enable
0: No interrupt generation
1: An interrupt is generated if the embedded synchronization codes are not received in the correct order.
Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_IE:** Overrun interrupt enable
0: No interrupt generation
1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received.

Bit 0 **FRAME_IE:** Capture complete interrupt enable
0: No interrupt generation
1: An interrupt is generated at the end of each received frame/crop window (in crop mode).

15.8.5 DCMI masked interrupt status register (DCMI_MIS)

This DCMI_MIS register is a read-only register. When read, it returns the current masked status value (depending on the value in DCMI_IER) of the corresponding interrupt. A bit in this register is set if the corresponding enable bit in DCMI_IER is set and the corresponding bit in DCMI_RIS is set.

Address offset: 0x10

Reset value: 0x0000

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

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 VSYNC transitions
1: An interrupt is generated on each VSYNC transition from the inactive to the active state and the VSYNC_IE bit is set in DCMI_IER.
The active state of the VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR_MIS:** Synchronization error masked interrupt status
This bit gives the status of the masked synchronization error interrupt
0: No interrupt is generated on a synchronization error
1: An interrupt is generated if the embedded synchronization codes are not received in the correct order and the ERR_IE bit in DCMI_IER is set.
Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_MIS:** Overrun masked interrupt status
This bit gives the status of the masked overflow interrupt
0: No interrupt is generated on overrun
1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received and the OVR_IE bit is set in DCMI_IER.

Bit 0 **FRAME_MIS:** Capture complete masked interrupt status
This bit gives the status of the masked capture complete interrupt
0: No interrupt is generated after a complete capture
1: An interrupt is generated at the end of each received frame/crop window (in crop mode) and the FRAME_IE bit is set in DCMI_IER.

15.8.6 DCMI interrupt clear register (DCMI_ICR)

Address offset: 0x14

Reset value: 0x0000 0x0000

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

The DCMI_ICR register is write-only. Writing a ‘1’ into a bit of this register clears the corresponding bit in the DCMI_RIS and DCMI_MIS registers. Writing a ‘0’ has no effect.

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **LINE_ISC:** line interrupt status clear
Writing a ‘1’ into this bit clears LINE_RIS in the DCMI_RIS register

Bit 3 **VSYNC_ISC:** Vertical synch interrupt status clear
Writing a ‘1’ into this bit clears the VSYNC_RIS bit in DCMI_RIS

Bit 2 **ERR_ISC:** Synchronization error interrupt status clear
Writing a ‘1’ into this bit clears the ERR_RIS bit in DCMI_RIS
Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_ISC:** Overrun interrupt status clear
Writing a ‘1’ into this bit clears the OVR_RIS bit in DCMI_RIS

Bits 0 **FRAME_ISC:** Capture complete interrupt status clear
Writing a ‘1’ into this bit clears the FRAME_RIS bit in DCMI_RIS

15.8.7 DCMI embedded synchronization code register (DCMI_ESCR)

Address offset: 0x18

Reset value: 0x0000 0x0000

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

Bits 31:24 **FEC:** Frame end delimiter code

This byte specifies the code of the frame end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FEC.

If FEC is programmed to 0xFF, all the unused codes (0xFF0000XY) are interpreted as frame end delimiters.

Bits 23:16 **LEC:** Line end delimiter code

This byte specifies the code of the line end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LEC.

Bits 15:8 **LSC:** Line start delimiter code

This byte specifies the code of the line start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LSC.

Bits 7:0 **FSC:** Frame start delimiter code

This byte specifies the code of the frame start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FSC.

If FSC is programmed to 0xFF, no frame start delimiter is detected. But, the 1st occurrence of LSC after an FEC code will be interpreted as a start of frame delimiter.

15.8.8 DCMI embedded synchronization unmask register (DCMI_ESUR)

Address offset: 0x1C

Reset value: 0x0000 0x0000

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

Bits 31:24 **FEU:** Frame end delimiter unmask

This byte specifies the mask to be applied to the code of the frame end delimiter.

0: The corresponding bit in the FEC byte in DCMI_ESCR is masked while comparing the frame end delimiter with the received data.

1: The corresponding bit in the FEC byte in DCMI_ESCR is compared while comparing the frame end delimiter with the received data

Bits 23:16 **LEU:** Line end delimiter unmask

This byte specifies the mask to be applied to the code of the line end delimiter.

0: The corresponding bit in the LEC byte in DCMI_ESCR is masked while comparing the line end delimiter with the received data

1: The corresponding bit in the LEC byte in DCMI_ESCR is compared while comparing the line end delimiter with the received data

Bits 15:8 **LSU:** Line start delimiter unmask

This byte specifies the mask to be applied to the code of the line start delimiter.

0: The corresponding bit in the LSC byte in DCMI_ESCR is masked while comparing the line start delimiter with the received data

1: The corresponding bit in the LSC byte in DCMI_ESCR is compared while comparing the line start delimiter with the received data

Bits 7:0 **FSU:** Frame start delimiter unmask

This byte specifies the mask to be applied to the code of the frame start delimiter.

0: The corresponding bit in the FSC byte in DCMI_ESCR is masked while comparing the frame start delimiter with the received data

1: The corresponding bit in the FSC byte in DCMI_ESCR is compared while comparing the frame start delimiter with the received data

15.8.9 DCMI crop window start (DCMI_CWSTRT)

Address offset: 0x20

Reset value: 0x0000 0x0000

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

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.

Bit 13:0 **HOFFCNT[13:0]:** Horizontal offset count

This value gives the number of pixel clocks to count before starting a capture.

15.8.10 DCMI crop window size (DCMI_CWSIZE)

Address offset: 0x24

Reset value: 0x0000 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		VLINE13:0]														Reserved		CAPCNT[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bitst 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
....

15.8.11 DCMI data register (DCMI_DR)

Address offset: 0x28

Reset value: 0x0000 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte3								Byte2								Byte1								Byte0							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Data byte 3

Bits 23:16 Data byte 2

Bits 15:8 Data byte 1

Bits 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.

15.8.12 DCMI register map

Table 87 summarizes the DCMI registers.

Table 87. DCMI register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DCMI_CR	Reserved																	ENABLE	Reserved		EDM		FCRC		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
	Reset value																		0			0	0	0	0	0	0	0	0	0	0	0	0
0x04	DCMI_SR	Reserved																													FNE	VSYNC	HSYNC
	Reset value																														0	0	0
0x08	DCMI_RIS	Reserved																											LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
	Reset value																												0	0	0	0	0
0x0C	DCMI_IER	Reserved																											LINE_IE	VSYNC_IE	ERR_IE	OVR_IE	FRAME_IE
	Reset value																												0	0	0	0	0
0x10	DCMI_MIS	Reserved																											LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS
	Reset value																												0	0	0	0	0

Table 87. DCMI register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x14	DCMI_ICR	Reserved																											LINE_ISC	VSYNC_ISC	ERR_ISC	OVR_ISC	FRAME_ISC
	Reset value																												0	0	0	0	0
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
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
0x20	DCMI_CWSTRT	Reserved			VST[12:0													Reserved		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
0x24	DCMI_CWSIZE	Reserved		VLINE13:0]														Reserved		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
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

Refer to *Section 2.3: Memory map* for the register boundary addresses.

16 LCD-TFT controller (LTDC)

This section applies only to STM32F429xx/439xx devices.

16.1 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 synchronisation, Pixel Clock and Data Enable as output to interface directly to a variety of LCD and TFT panels.

16.2 LTDC main features

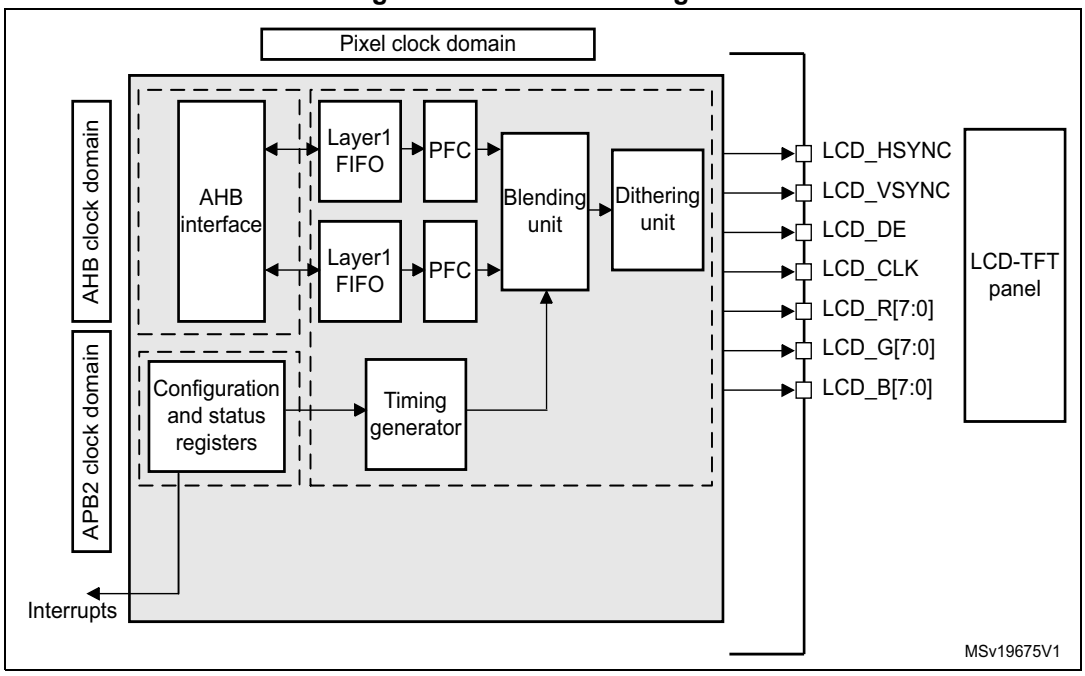
- 24-bit RGB Parallel Pixel Output; 8 bits-per-pixel (RGB888)
- 2 display layers with dedicated FIFO (64x32-bit)
- Color Look-Up Table (CLUT) up to 256 color (256x24-bit) per layer
- Supports up to XGA (1024x768) resolution
- Programmable timings for different display panels
- Programmable Background 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

16.3 LTDC functional description

16.3.1 LTDC block diagram

The block diagram of the LTDC is shown in *Figure 81: LTDC block diagram*.

Figure 81. LTDC block diagram



Layer FIFO: One FIFO 64x32 bit per layer.

PFC: Pixel Format Convertor 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 16.4.1* and *Section 16.4.2*.

16.3.2 LTDC reset and clocks

The LCD-TFT controller peripheral uses 3 clock domains:

- The AHB clock domain (HCLK) is used for data transfer from the memories to the Layer FIFO and frame buffer configuration register
- The APB2 clock domain (PCLK2) is used for global configuration register and interrupt registers
- The Pixel Clock domain (LCD_CLK) is used to generate LCD-TFT interface signals, pixel data generation and layer configuration. The LCD_CLK output should be configured following the panel requirements. The LCD_CLK is configured through the PLLSAI (refer to RCC section).

Table 88 summarizes the clock domain for each register.

Table 88. LTDC registers versus clock domain

LTDC registers	Clock domain
LTDC_LxCR	HCLK
LTDC_LxCFBAR	
LTDC_LxCFBLR	
LTDC_LxCFBLNR	
LTDC_SRCR	PCLK2
LTDC_IER	
LTDC_ISR	
LTDC_ICR	
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 when accessing the LTDC registers since the APB2 bus is stalling when the following operations are ongoing:

- Register write access and update for 6 xPCKL2 period + 5x LCD_CLK period (5x HCLK period for register on AHB clock domain)
- Register read access for 7xPCKL2 period + 5x LCD_CLK period (5x HCLK period for register on AHB clock domain).

For registers on PCLK2 clock domain, APB2 bus is stalling during the register write access for 6 xPCKL2 period and 7xPCKL2 period for read access.

The LCD controller can be reset by setting the corresponding bit in the RCC_APB2RSTR register. It resets the three clock domains.

16.3.3 LCD-TFT pins and signal interface

The Table below summarizes the LTDC signal interface:

Table 89. LCD-TFT 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
LCD_G[7:0]	O	Data: 8-bit Green data
LCD_B[7:0]	O	Data: 8-bit Blue data

The LCD-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 8bpp are used to output for example RGB565 or RGB666 to interface on 16b-bit 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].

16.4 LTDC programmable parameters

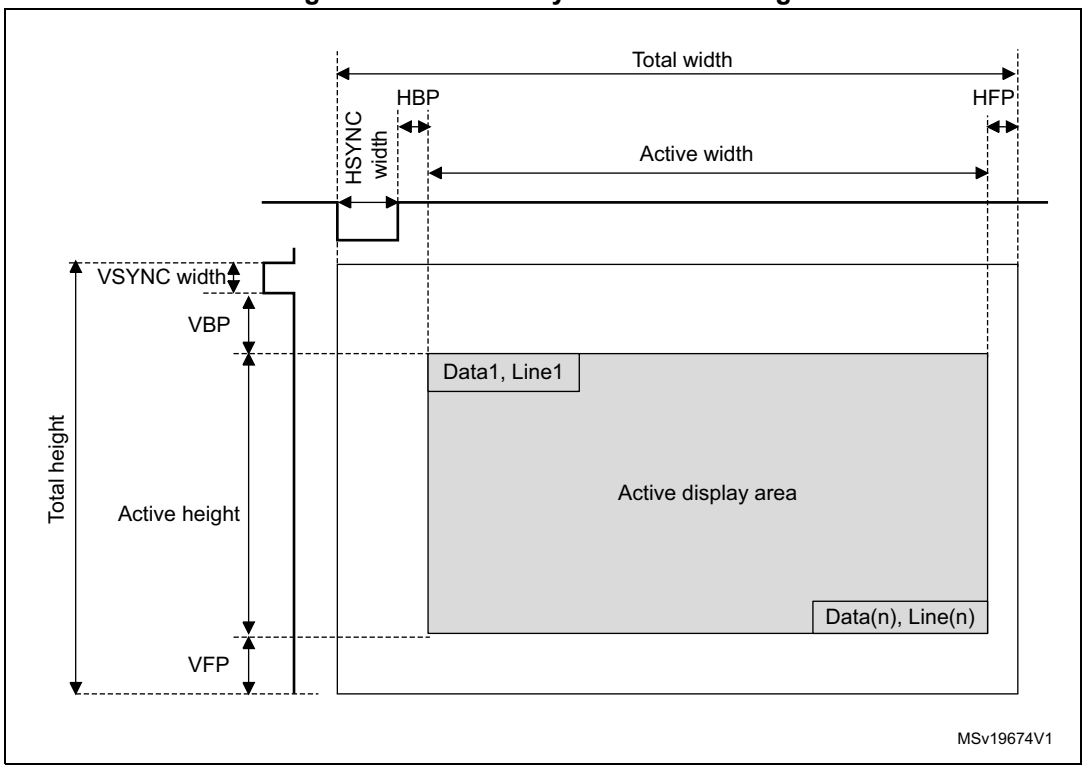
The LCD-TFT controller provides flexible configurable parameters. It can be enabled or disabled through the **LTDC_GCR** register.

16.4.1 LTDC Global configuration parameters

Synchronous Timings:

Figure 82 presents the configurable timing parameters generated by the Synchronous Timings Generator block presented in the block diagram *Figure 81*. It generates the Horizontal and Vertical Synchronization timings panel signals, the Pixel Clock and the Data Enable signals.

Figure 82. 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 (should be extracted from Panel datasheet):

- Horizontal and Vertical Synchronization width: 0x8 pixels and 0x4 lines
- Horizontal and Vertical back porch: 0x7 pixels and 0x2 lines
- Active Width and Active Height: 0x280 pixels, 0x1E0 lines (640x480)
- Horizontal front porch: 0x6 pixels
- Vertical front porch: 0x2 lines

The programmed values in the LTDC Timings registers will be:

- **LTDC_SSCR** register: to be programmed to 0x00070003. (HSW[11:0] is 0x7 and VSH[10:0] is 0x3)
- **LTDC_BPCR** register: to be programmed to 0x000E0005. (AHBP[11:0] is 0xE(0x8 + 0x6) and AVBP[10:0] is 0x5(0x4 + 0x1))
- **LTDC_AWCR** register: to be programmed to 0x028E01E5. (AAW[11:0] is 0x28E(0x8 +0x7 +0x27F) and AAH[10:0] is 0x1E5(0x4 +0x2 + 0x1DF)
- **LTDC_TWCR** register: to be programmed to 0x00000294. (TOTALW[11:0] is 0x294(0x8 +0x7 +0x280 + 0x5)
- **LTDC_THCR** register: to be programmed to 0x000001E7. (TOTALH[10:0] is 0x1E7(0x4 +0x2 + 0x1E0 + 1)

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 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 Dither 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 >= 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 Layer 1 and Layer 2 registers except the **LTDC_LxCLUTWR** register.

Interrupt generation event

Refer to *Section 16.5: LTDC interrupts* for interrupt configuration.

16.4.2 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

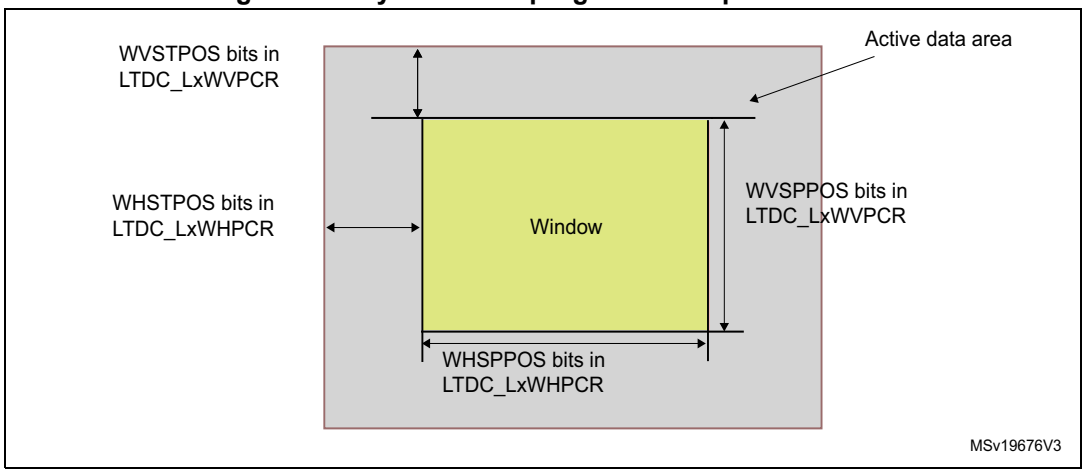
Every layer can be positioned and resized and it must be inside the Active Display area.

The window position and size are configured through the top-left and bottom-right X/Y positions and the Internal timing generator which 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 83*

- 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 83. 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 which have 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)

The figure below describes the pixel data mapping depending on the selected format.

Table 90. Pixel Data mapping versus Color Format

ARGB8888			
@+3 A_x[7:0]	@+2 R_x[7:0]	@+1 G_x[7:0]	@ B_x[7:0]
@+7 A_{x+1}[7:0]	@+6 R_{x+1}[7:0]	@+5 G_{x+1}[7:0]	@+4 B_{x+1}[7:0]
RGB888			
@+3 B_{x+1}[7:0]	@+2 R_x[7:0]	@+1 G_x[7:0]	@ B_x[7:0]
@+7 G_{x+2}[7:0]	@+6 B_{x+2}[7:0]	@+5 R_{x+1}[7:0]	@+4 G_{x+1}[7:0]
RGB565			
@+3 R_{x+1}[4:0] G_{x+1}[5:3]	@+2 G_{x+1}[2:0] B_{x+1}[4:0]	@+1 R_x[4:0] G_x[5:3]	@ G_x[2:0] B_x[4:0]

Table 90. Pixel Data mapping versus Color Format (continued)

ARGB8888			
@+7 R_{x+3}[4:0] G_{x+3}[5:3]	@+6 G_{x+3}[2:0] B_{x+3}[4:0]	@+5 R_{x+2}[4:0] G_{x+2}[5:3]	@+4 G_{x+2}[2:0] B_{x+2}[4:0]
ARGB1555			
@+3 A_{x+1}[0]R_{x+1}[4:0] G_{x+1}[4:3]	@+2 G_{x+1}[2:0] B_{x+1}[4:0]	@+1 A_x[0] R_x[4:0] G_x[4:3]	@ G_x[2:0] B_x[4:0]
@+7 A_{x+3}[0]R_{x+3}[4:0] G_{x+3}[4:3]	@+6 G_{x+3}[2:0] B_{x+3}[4:0]	@+5 A_{x+2}[0]R_{x+2}[4:0]G_{x+2}[4:3]	@+4 G_{x+2}[2:0] B_{x+2}[4:0]
ARGB4444			
@+3 A_{x+1}[3:0]R_{x+1}[3:0]	@+2 G_{x+1}[3:0] B_{x+1}[3:0]	@+1 A_x[3:0] R_x[3:0]	@ G_x[3:0] B_x[3:0]
@+7 A_{x+3}[3:0]R_{x+3}[3:0]	@+6 G_{x+3}[3:0] B_{x+3}[3:0]	@+5 A_{x+2}[3:0]R_{x+2}[3:0]	@+4 G_{x+2}[3:0] B_{x+2}[3:0]
L8			
@+3 L_{x+3}[7:0]	@+2 L_{x+2}[7:0]	@+1 L_{x+1}[7:0]	@ L_x[7:0]
@+7 L_{x+7}[7:0]	@+6 L_{x+6}[7:0]	@+5 L_{x+5}[7:0]	@+4 L_{x+4}[7:0]
AL44			
@+3 A_{x+3}[3:0] L_{x+3}[3:0]	@+2 A_{x+2}[3:0] L_{x+2}[3:0]	@+1 A_{x+1}[3:0] L_{x+1}[3:0]	@ A_x[3:0] L_x[3:0]
@+7 A_{x+7}[3:0] L_{x+7}[3:0]	@+6 A_{x+6}[3:0] L_{x+6}[3:0]	@+5 A_{x+5}[3:0] L_{x+5}[3:0]	@+4 A_{x+4}[3:0] L_{x+4}[3:0]
AL88			
@+3 A_{x+1}[7:0]	@+2 L_{x+1}[7:0]	@+1 A_x[7:0]	@ L_x[7:0]
@+7 A_{x+3}[7:0]	@+6 L_{x+3}[7:0]	@+5 A_{x+2}[7:0]	@+4 L_{x+2}[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 will 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

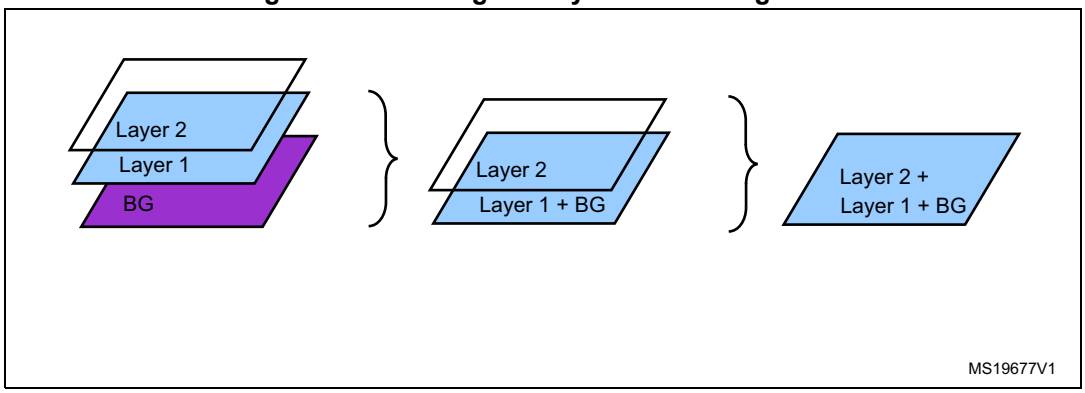
Every layer has a configurable pitch for the color frame buffer, which is the distance between the start of one line and the beginning of the next line in bytes. It is configured through the **LTDC_LxCFBLR** register.

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 84*.

Figure 84. Blending two layers with background



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 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.

16.5 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 the figure below.

Figure 85. Interrupt events

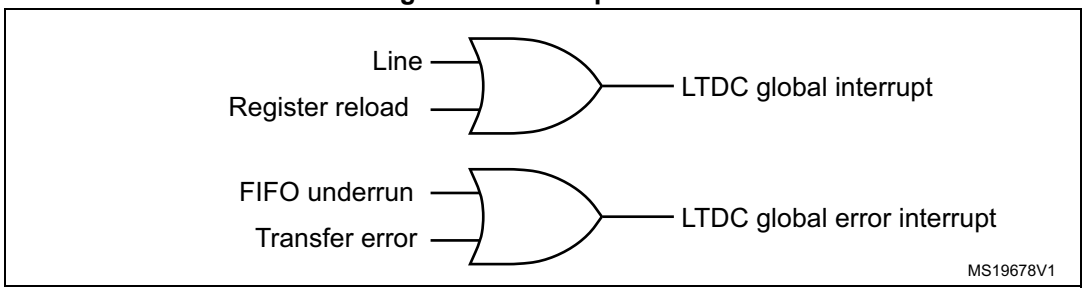


Table 91. LTDC interrupt requests

Interrupt event	Event flag	Enable Control bit
Line	LIF	LIE
Register Reload	RRIF	RRIEN
FIFO Underrun	FUDERRIF	FUDERRIE
Transfer Error	TERRIF	TERRIE

16.6 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 16.4.1: 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.
  - The pixel input format in the **LTDC_LxPFCR** register
  - The color frame buffer start address in the **LTDC_LxCFBAR** register
  - The line length and pitch of the color frame buffer in the **LTDC_LxCFBLR** register
  - The number of lines of the color frame buffer in the **LTDC_LxCFBLNR** register
  - if needed, load the CLUT with the RGB values and its address in the **LTDC_LxCLUTWR** register
  - If needed, configure 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, dithering and color keying can be enabled respectively in the **LTDC_GCR and LTDC_LxCKCR** registers. It 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 should not be modified again before the reload has been done. Thus, a new write to the same register will override the previous configuration if not yet reloaded.

16.7 LTDC registers

16.7.1 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 82* and *Section 16.4: 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
Reserved				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
Reserved				VSH[10:0]											
				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 **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.

16.7.2 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 82* and *Section 16.4: 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
Reserved				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
Reserved				AVBP[10:0]											
				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 **AHBP[11:0]**: Accumulated Horizontal back porch (in units of pixel clock period)
These bits define the Accumulated Horizontal back porch width which 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 which 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.

16.7.3 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 82* and *Section 16.4: 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
Reserved				AAW[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
Reserved				AAH[10:0]											
				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 **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. The maximum Active Width supported is 0x400.

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. The maximum Active Height supported is 0x300.

16.7.4 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 82* and *Section 16.4: 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
Reserved				TOTALW[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

16	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				TOTALH[10:0]											
				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 **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.

16.7.5 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	Reserved											DEN
rw	rw	rw	rw												rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DRW[2:0]			Reserved	DGW[2:0]			Reserved	DBW[2:0]			Reserved			LTDCEN
	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**: Data Enable Polarity
This bit is set and cleared by software.
0: Data Enable polarity is active low
1: Data Enable polarity is active high

Bit 28 **PCPOL**: Pixel Clock Polarity
This bit is set and cleared by software.
0: input pixel clock
1: inverted input pixel clock

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 bit
This bit is set and cleared by software.
0: LTDC disable
1: LTDC enable

16.7.6 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														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 will be read.

16.7.7 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
Reserved								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

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

16.7.8 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												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

16.7.9 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												RRIF	TERRIF	FUIF	LIF
												r	r	r	r

Bits 31:24 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: A FIFO underrun interrupt is 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: A Line interrupt is generated, when a programmed line is reached

16.7.10 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CRRIF	CTERRIF	CFUIF	CLIF
												w	w	w	w

Bits 31:24 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.

16.7.11 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 82*.

Address offset: 0x40

Reset value: 0x0000 0000

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

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

16.7.12 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

16.7.13 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												HSYNCS	VSYNCS	HDES	VDES
												r	r	r	r

Bits 31:24 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.*

16.7.14 LTDC Layerx Control Register (LTDC_LxCR) (where x=1..2)

Address offset: 0x84 + 0x80 x (*Layerx* -1), *Layerx* = 1 or 2

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											CLUTEN	Reserved		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) on page 488*

Bit 3 Reserved, must be kept at reset value

Bit 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

16.7.15 LTDC Layerx Window Horizontal Position Configuration Register (LTDC_LxWHPCR) (where x=1..2)

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[10:0] 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 (*Layerx* -1), *Layerx* = 1 or 2

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				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
Reserved				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.
The following condition must be respected:
WHSPPOS[11:0] ≥ AHBP[10:0] 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.
The following condition must be respected:
WHSTPOS[11:0] must be ≤ AAW[10:0] bits (programmed in the 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] should be programmed to 0x14 (0xE+1+0x5)
2. Layer window last pixel: WHSPPOS[11:0] should be programmed to 0x28A

16.7.16 LTDC Layerx Window Vertical Position Configuration Register (LTDC_LxWVPCR) (where x=1..2)

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 (*Layerx* -1), *Layerx* = 1 or 2

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					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
Reserved					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 configures the last visible line of the layer window.
The following condition must be respected:
WHSPPOS[11:0] must be ≥ 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.
The following condition must be respected:
WHSTPOS[11:0] must be ≤ AAH[10:0] bits (programmed in the 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] should be programmed to 0xE (0x5 + 1 + 0x8)
2. Layer window last line: WVSPPOS[10:0] should be programmed to 0x1DA

16.7.17 LTDC Layerx Color Keying Configuration Register (LTDC_LxCKCR) (where x=1..2)

This register defines the color key value (RGB), which is used by the Color Keying.

Address offset: 0x90 + 0x80 x (*Layerx* -1), *Layerx* = 1 or 2

Reset value: 0x0000 0000

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

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

16.7.18 LTDC Layerx Pixel Format Configuration Register (LTDC_LxPFCR) (where x=1..2)

This register defines the pixel format which 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 (*Layerx* -1), *Layerx* = 1 or 2

Reset value: 0x0000 0000

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

16.7.19 LTDC Layerx Constant Alpha Configuration Register (LTDC_LxCACR) (where x=1..2)

This register defines the constant alpha value (divided by 255 by Hardware), which is used in the alpha blending. Refer to LTDC_LxBFCR register.

Address offset: 0x98 + 0x80 x (*Layerx* -1), *Layerx* = 1 or 2

Reset value: (Layerx -1) 0x0000 00FF

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

16.7.20 LTDC Layerx Default Color Configuration Register (LTDC_LxDCCR) (where x=1..2)

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 (*Layerx* -1), *Layerx* = 1 or 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]							
rw								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCGREEN[7:0]								DCBLUE[7:0]							
rw								rw	rw	rw	rw	rw	rw	rw	rw

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

16.7.21 LTDC Layerx Blending Factors Configuration Register (LTDC_LxBFCR) (where x=1..2)

This register defines the blending factors F1 and F2.

The general blending formula is: BC = BF1 x C + BF2 x 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 x (*Layerx* -1), *Layerx* = 1 or 2

Reset value: 0x0000 0607

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					BF1[2:0]			Reserved					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*

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 = Constant Alpha x C + *(1 - Constant Alpha*) x Cs = *0.94 x 128 + (1- 0.94) x 48 = 123.*

16.7.22 LTDC Layerx Color Frame Buffer Address Register (LTDC_LxCFBAR) (where x=1..2)

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 (*Layerx* -1), *Layerx* = 1 or 2

Reset value: 0x0000 0000

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

Bits 31:0 **CFBADD[31:0]**: Color Frame Buffer Start Address
These bits defines the color frame buffer start address.

16.7.23 LTDC Layerx Color Frame Buffer Length Register (LTDC_LxCFBLR) (where x=1..2)

This register defines the color frame buffer line length and pitch.

Address offset: 0xB0 + 0x80 x (*Layerx* -1), *Layerx* = 1 or 2

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			CFBP[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
Reserved			CFBLL[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset valuer

Bits 28:16 **CFBP[12:0]**: Color Frame Buffer Pitch in bytes
These bits define the pitch which 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 x 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 256x2=512 bytes), 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 320x3=960), where pitch = line length requires a value of 0x03C003C3 to be written into this register.

16.7.24 LTDC Layerx ColorFrame Buffer Line Number Register (LTDC_LxCFBLNR) (where x=1..2)

This register defines the number of lines in the color frame buffer.

Address offset: 0xB4 + 0x80 x (*Layerx* -1), *Layerx* = 1 or 2

Reset value: 0x0000 0000

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

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 which 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.

16.7.25 LTDC Layerx CLUT Write Register (LTDC_LxCLUTWR) (where x=1..2)

This register defines the CLUT address and the RGB value.

Address offset: 0xC4 + 0x80 x (*Layerx* -1), *Layerx* = 1 or 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.

16.7.26 LTDC register map

The following table summarizes the LTDC registers. Refer to the register boundary addresses table for the LTDC register base address.

Table 92. LTDC 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
0x0008	LTDC_SSCR	Reserved						HSW[9:0]										Reserved					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
0x000C	LTDC_BPCR	Reserved						AHBP[11:0]										Reserved					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
0x0010	LTDC_AWCR	Reserved						AAV[11:0]										Reserved					AAH[10:0]										
	Reset value							0	9	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x0014	LTDC_TWCR	Reserved						TOTALW[11:0]										Reserved					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
0x0018	LTDC_GCR	HSPOL	VSPOL	DEPOL	PCPOL	Reserve											DEN	Reserved	DRW[2:0]			Reserved	DGW[2:0]			Reserved	DBW[2:0]			Reserved				LTDCEN
	Reset value	0	0	0	0												0		0	1	0		0	1	0		0	1	0				0
0x0024	LTDC_SRCR	Reserved																														VBR	IMR
	Reset value																															0	0
0x002C	LTDC_BCCR	Reserved								BC[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
0x0034	LTDC_IER	Reserved																												RRIE	TERRIE	FUIE	LIE
	Reset value																													0	0	0	0
0x0038	LTDC_ISR	Reserved																												RRIF	TERRIF	FUIF	LIF
	Reset value																													0	0	0	0
0x003C	LTDC_ICR	Reserved																												CRRIF	CTERRIF	CFUIF	CLIF
	Reset value																													0	0	0	0
0x0040	LTDC_LIPCR	Reserved																					LIPOS[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	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
0x0048	LTDC_CDSR	Reserved																												HSYNCS	VSYNCS	HDES	VDES
	Reset value																													1	1	1	1
0x0084	LTDC_L1CR	Reserved																										CLUTEN	Reserved			COLKEN	LEN
	Reset value																											0				0	0
0x0088	LTDC_L1WHPCR	Reserved				WHSPPOS[11:0]												Reserved	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

Table 92. LTDC 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
0x008C	LTDC_L1WVPCR	Reserved					WVSPPOS[10:0]											Reserved					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
0x0090	LTDC_L1CKCR	Reserved								CKRED[7:0]								CKGREEN[7:0]								CKBLUE[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
0x0094	LTDC_L1PFCR	Reserved																													PF[2:0]		
	Reset value																														0	0	0
0x0098	LTDC_L1CACR	Reserved																													CONSTA[7:0]		
	Reset value																														1	1	1
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	Reserved																					BF1[2:0]			Reserved					BF2[2:0]		
	Reset value																						1	1	0						1	1	1
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	Reserved			CFBP[12:0]													Reserved			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
0x00B4	LTDC_L1CFBLNR	Reserved																					CFBLNBR[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
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	Reserved																											CLUTEN	Reserved		COLKEN	LEN
	Reset value																												0			0	0
0x0108	LTDC_L2WHPCR	Reserved			WHSPPOS[11:0]													Reserved			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
0x010C	LTDC_L2WVPCR	Reserved					WVSPPOS[10:0]											Reserved					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
0x0110	LTDC_L2CKCR	Reserved								CKRED[7:0]								CKGREEN[7:0]								CKBLUE[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
0x0114	LTDC_L2PFCR	Reserved																													PF[2:0]		
	Reset value																														0	0	0
0x0118	LTDC_L2CACR	Reserved																													CONSTA[7:0]		
	Reset value																														1	1	1
0x011C	LTDC_L2DCCR	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
0x0120	LTDC_L2BFCR	Reserved																					BF1[2:0]			Reserved					BF2[2:0]		
	Reset value																						1	1	0						1	1	1
0x012C	LTDC_L2CFBAR	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
0x0130	LTDC_L2CFBLR	Reserved			CFBP[12:0]													Reserved			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

Table 92. LTDC 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
0x0134	LTDC_L2CFBLNR	Reserved																					CFBLNBR[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x0144	LTDC_L2CLUTWR	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

17 Advanced-control timers (TIM1 and TIM8)

This section applies to the whole STM32F4xx family, unless otherwise specified.

17.1 TIM1 and TIM8 introduction

The advanced-control timers (TIM1 and 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.

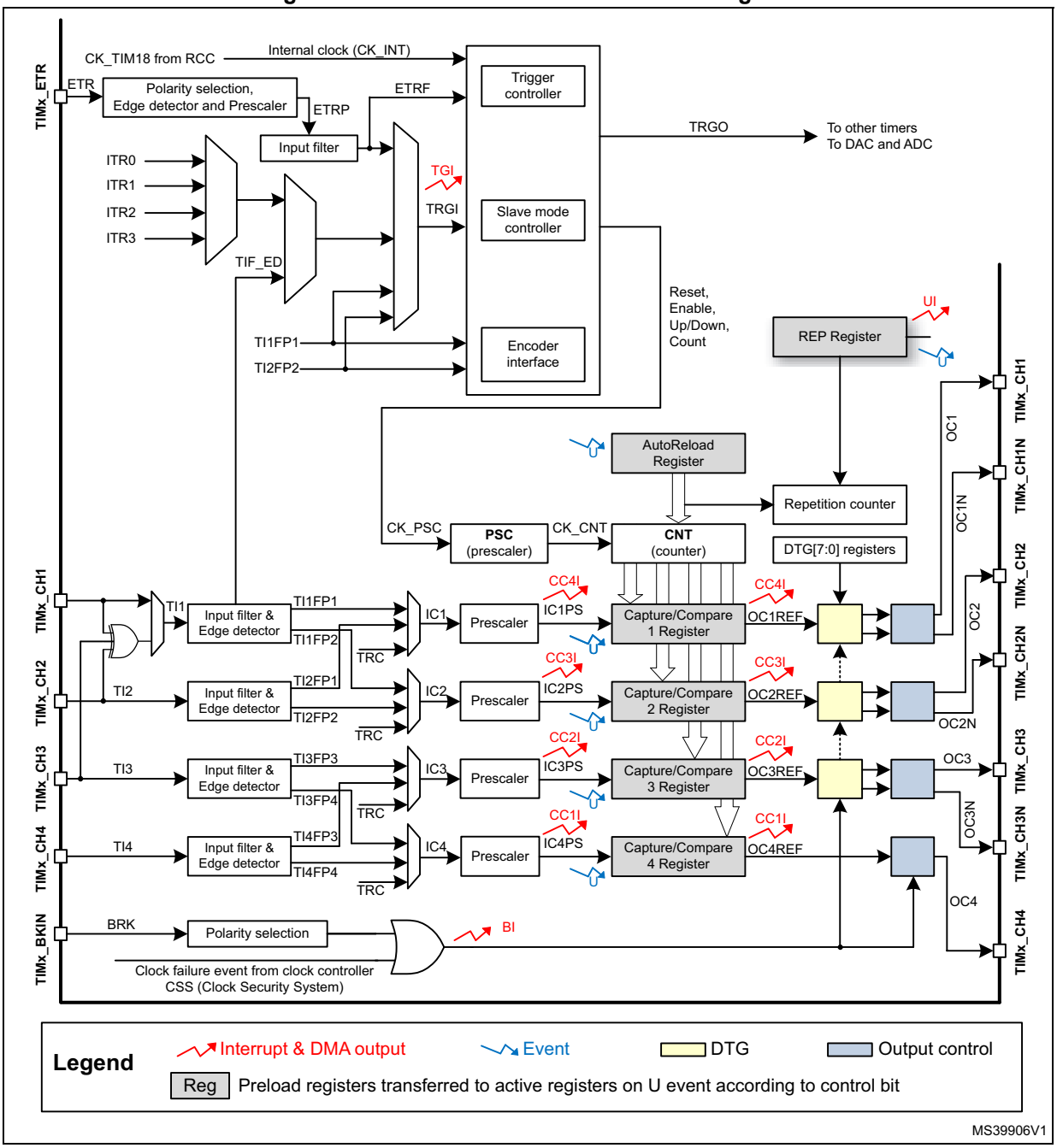
The advanced-control (TIM1 and TIM8) and general-purpose (TIMx) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 17.3.20*.

17.2 TIM1 and TIM8 main features

TIM1 and 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 4 independent channels for:
  - Input Capture
  - Output Compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- Break input to put the timer’s output signals in reset state or in a known state.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 86. Advanced-control timer block diagram



17.3 TIM1 and TIM8 functional description

17.3.1 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 87 and *Figure 88* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 87. Counter timing diagram with prescaler division change from 1 to 2

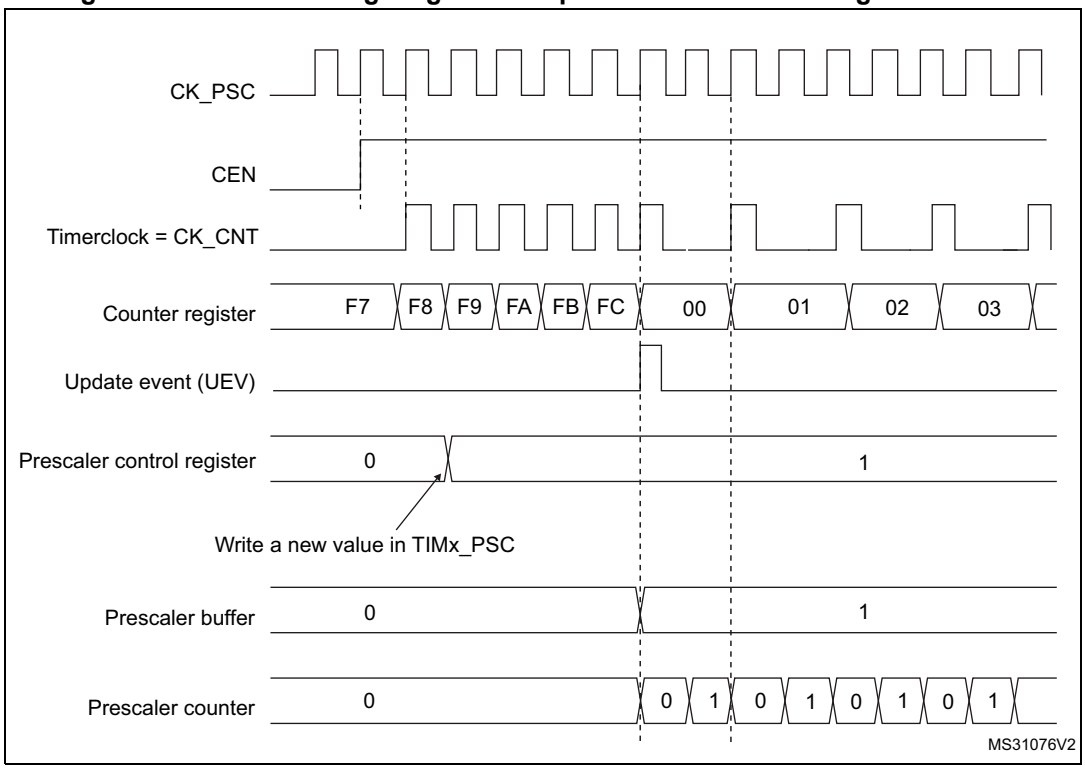
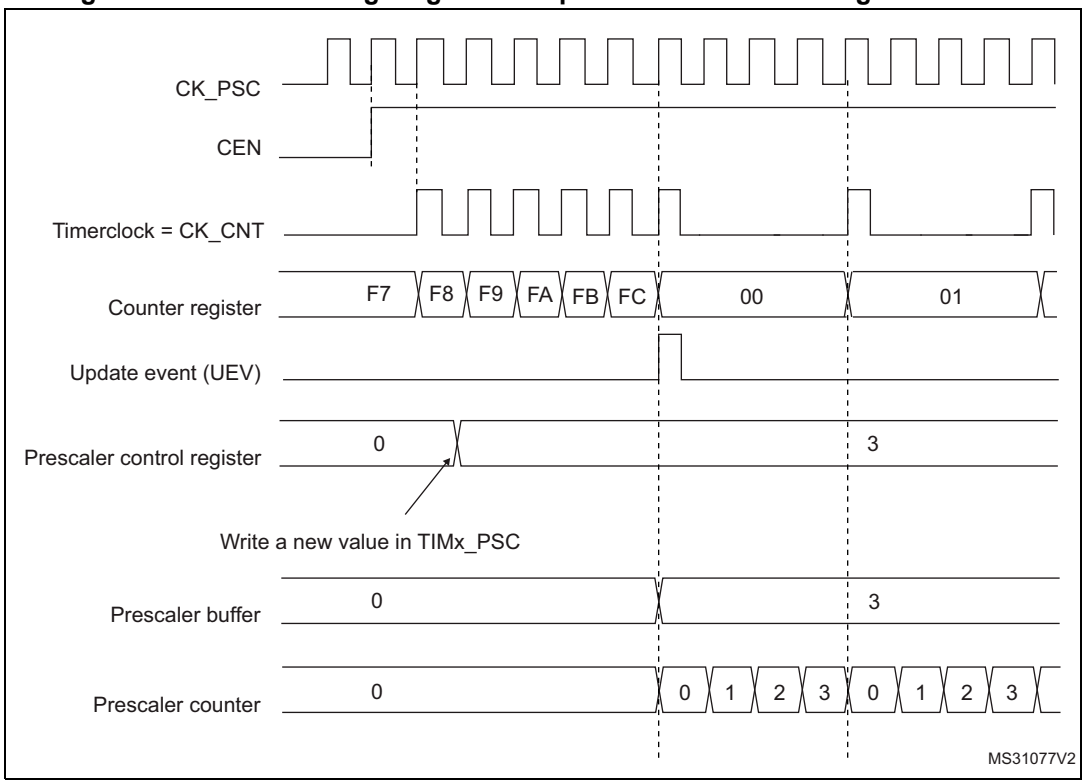


Figure 88. Counter timing diagram with prescaler division change from 1 to 4



17.3.2 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 plus one (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 89. Counter timing diagram, internal clock divided by 1

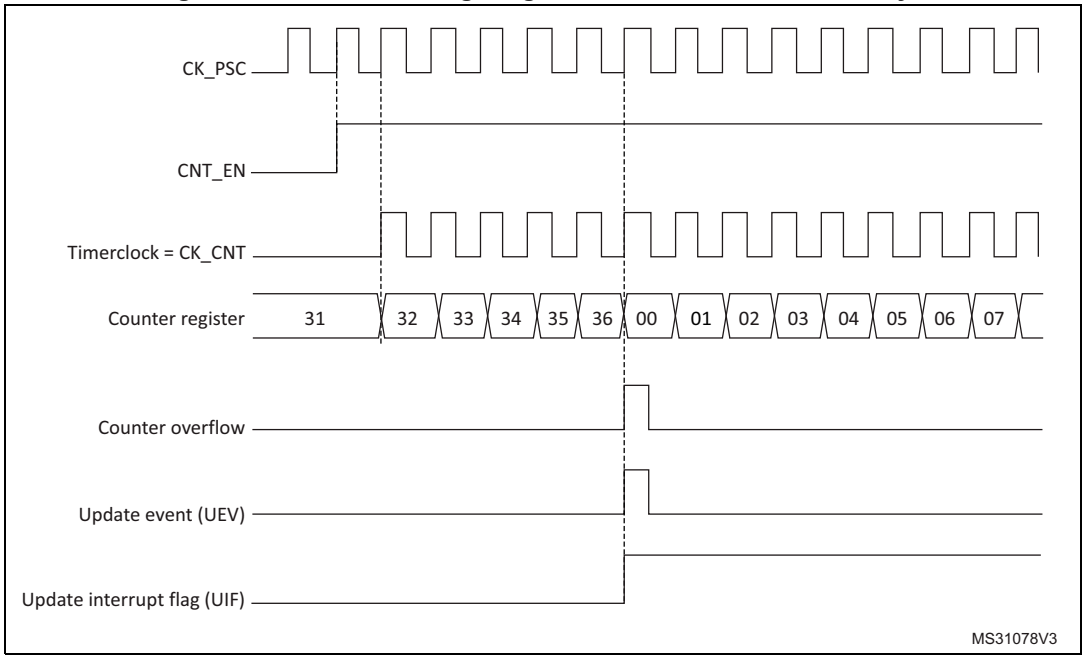


Figure 90. Counter timing diagram, internal clock divided by 2

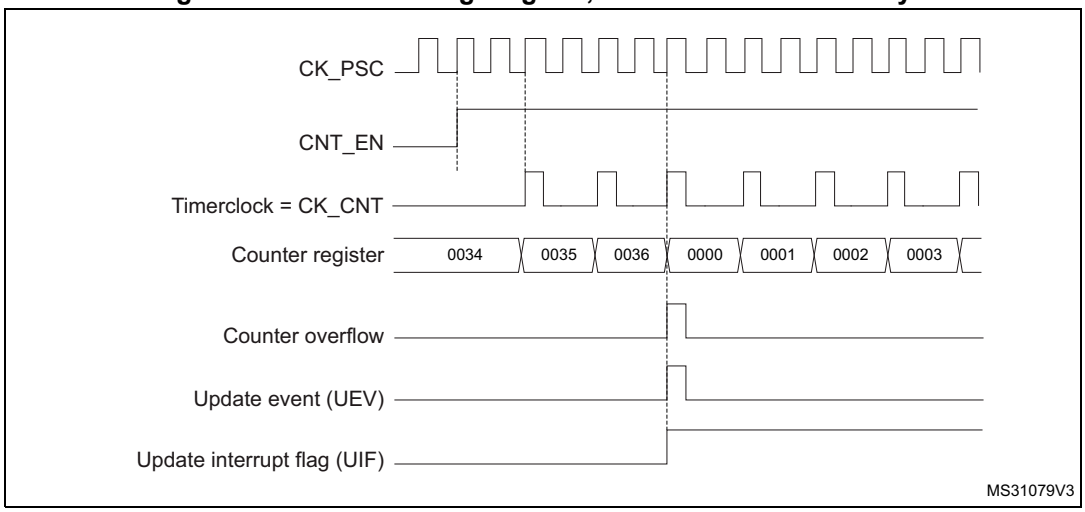


Figure 91. Counter timing diagram, internal clock divided by 4

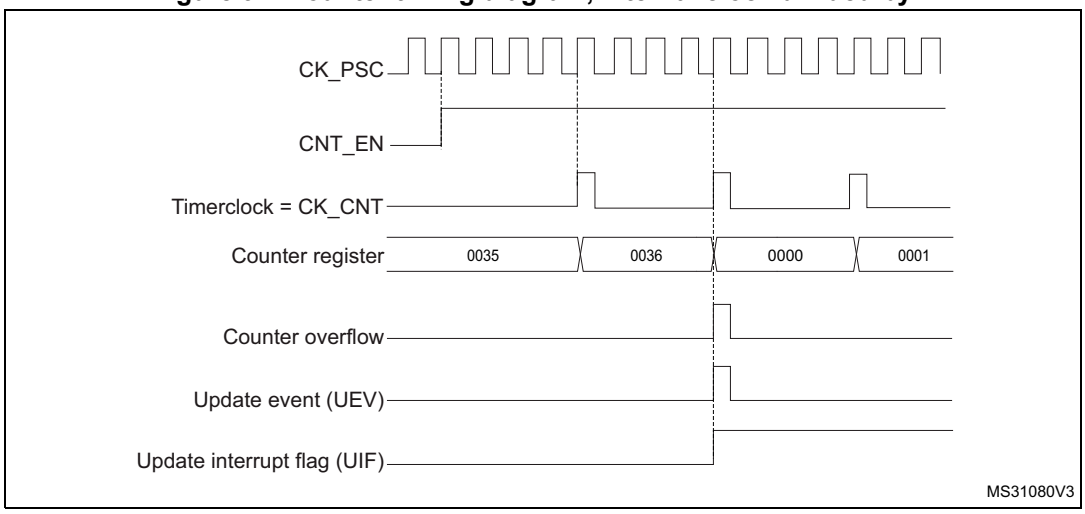


Figure 92. Counter timing diagram, internal clock divided by N

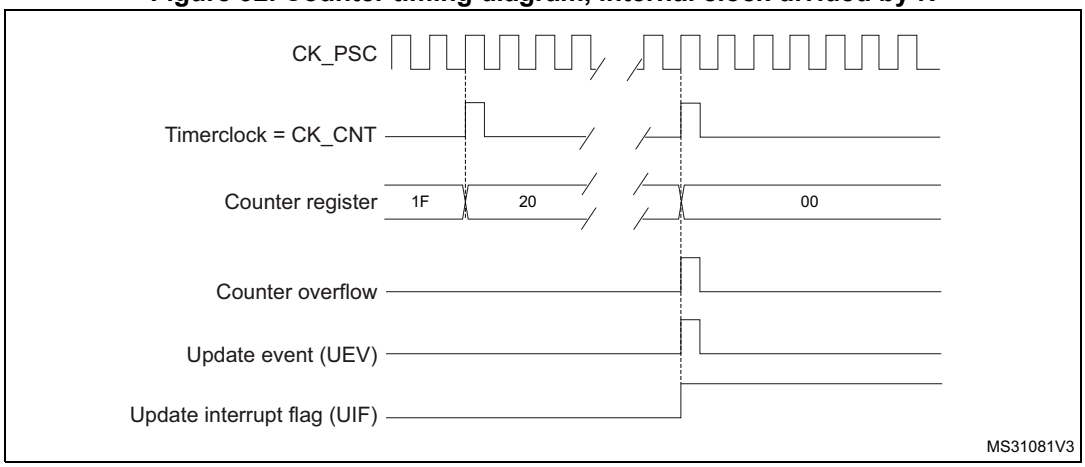


Figure 93. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

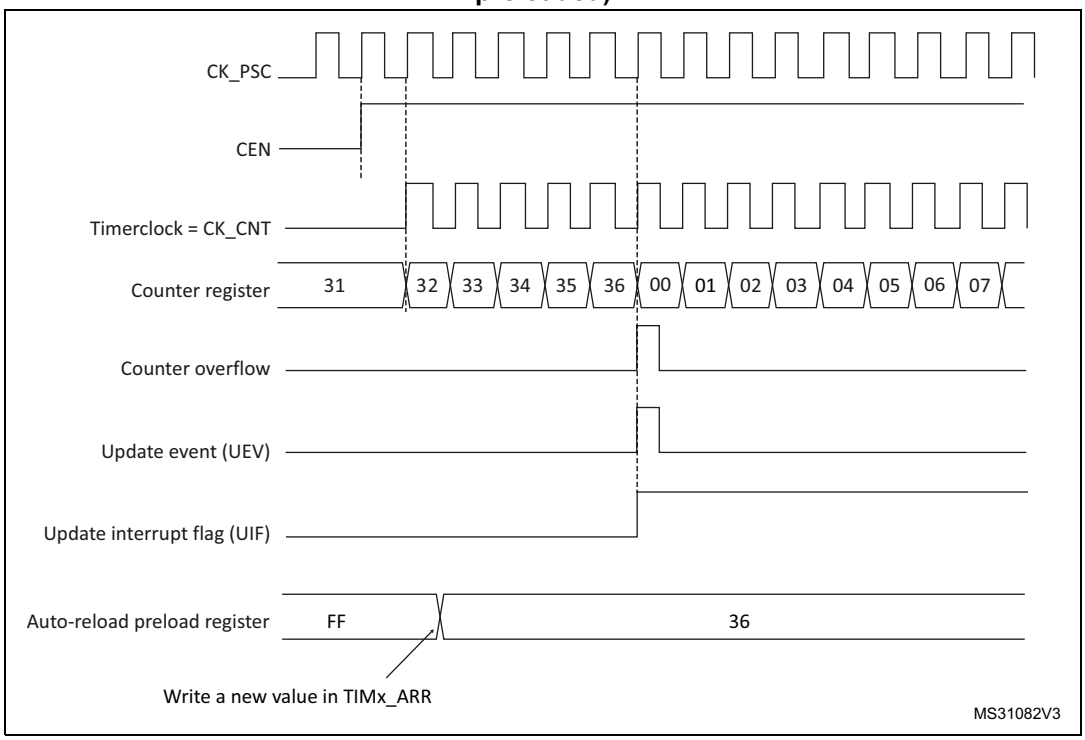
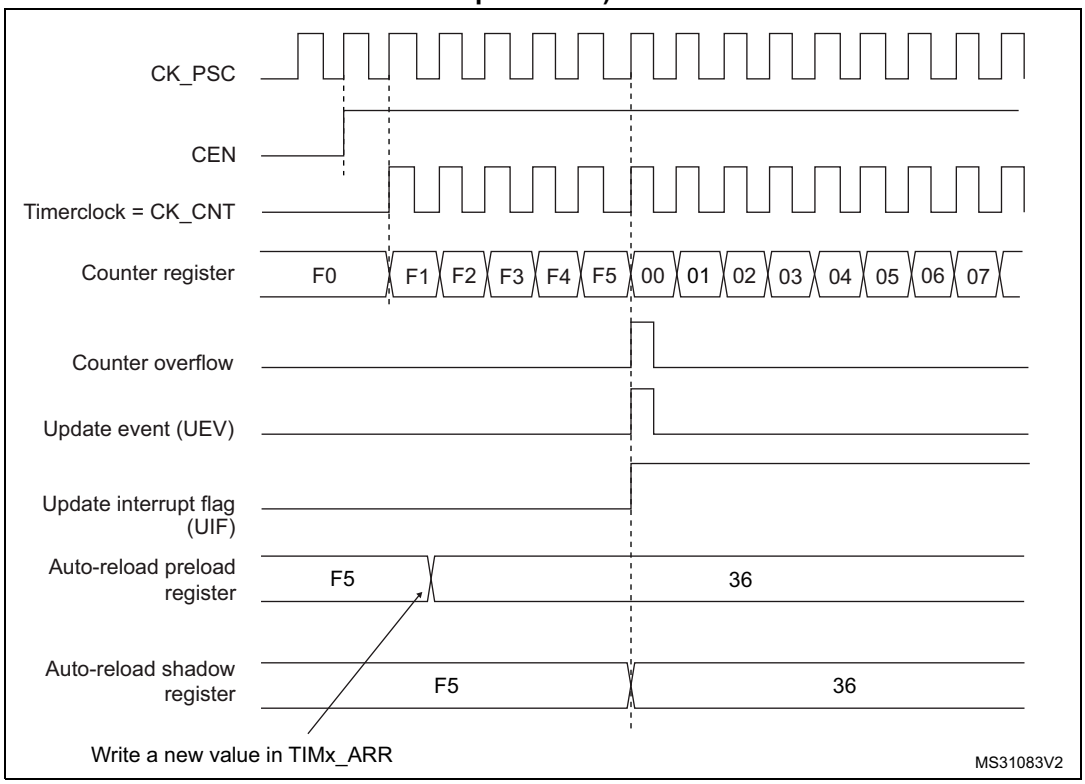


Figure 94. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



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 plus one (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 95. Counter timing diagram, internal clock divided by 1

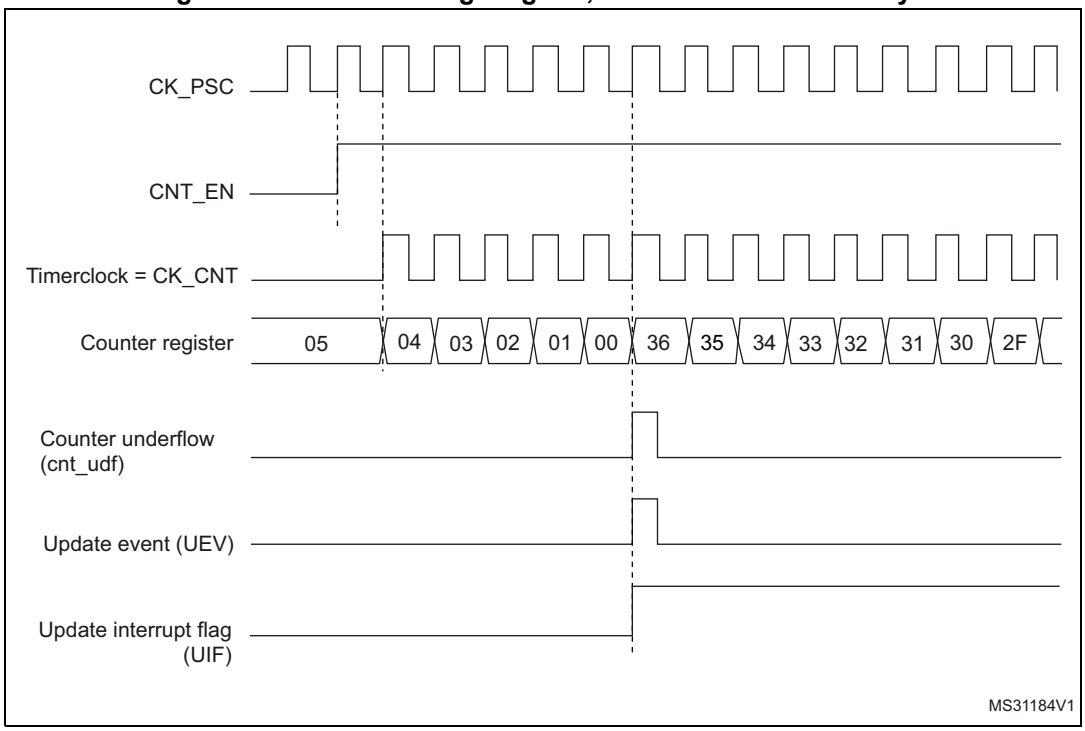


Figure 96. Counter timing diagram, internal clock divided by 2

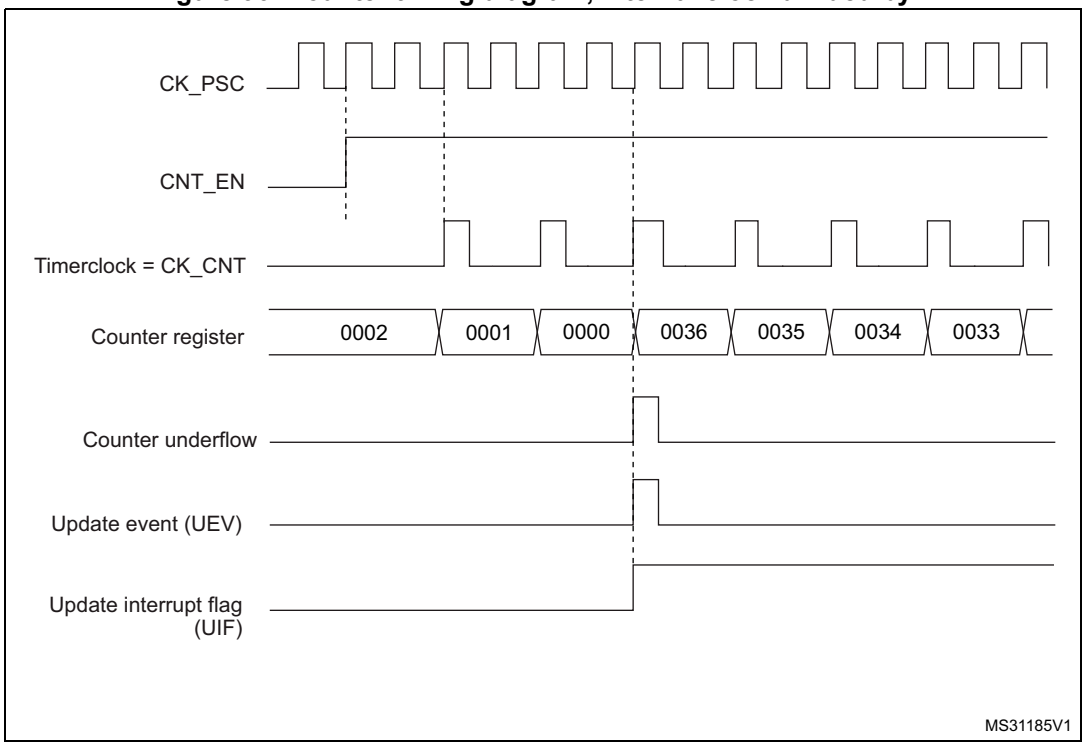


Figure 97. Counter timing diagram, internal clock divided by 4

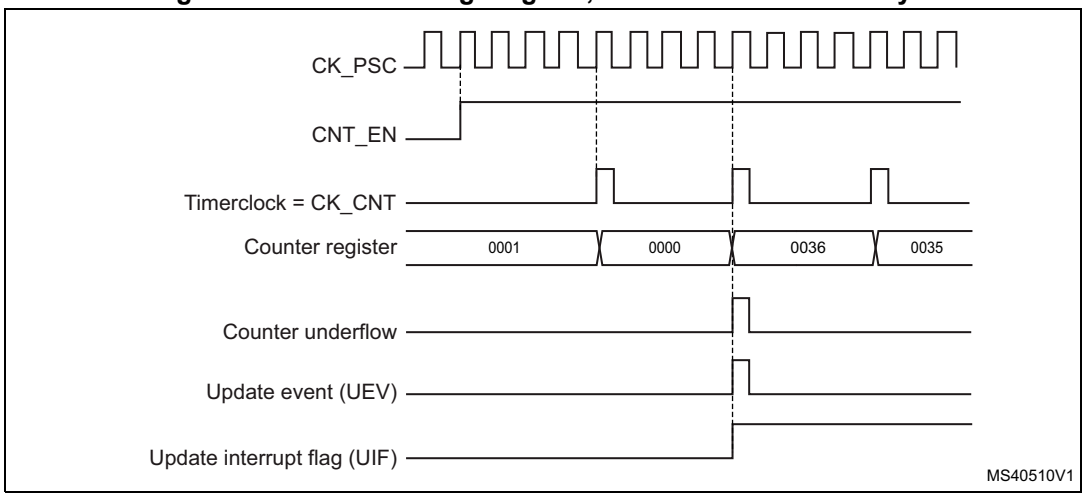


Figure 98. Counter timing diagram, internal clock divided by N

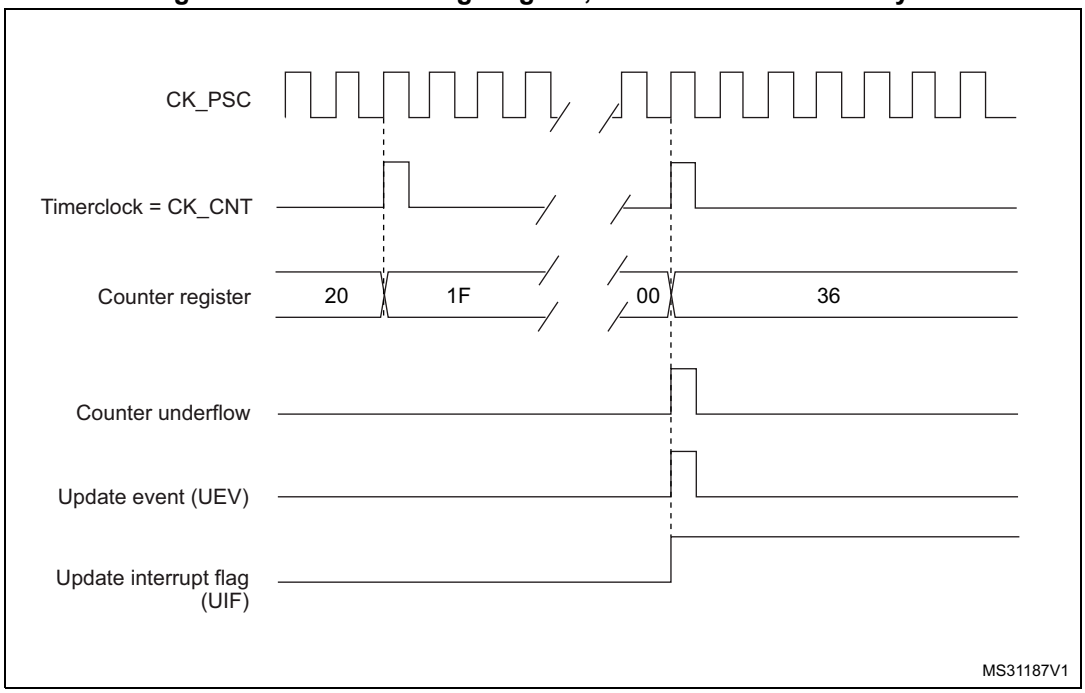
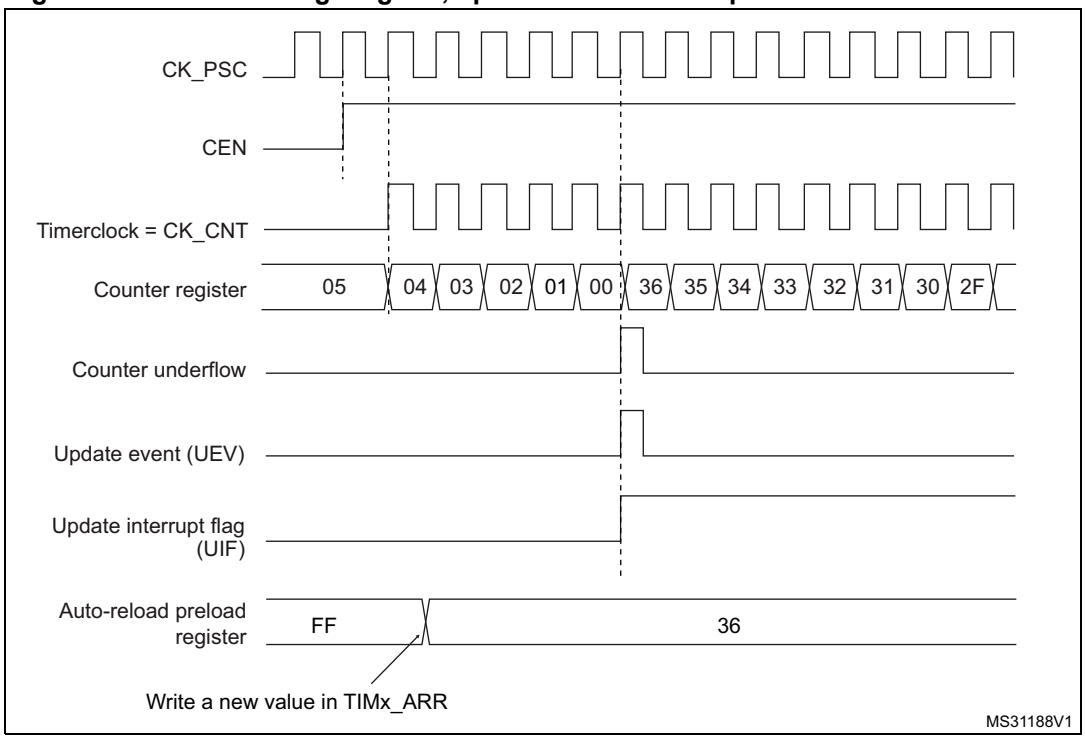


Figure 99. Counter timing diagram, update event when repetition counter is not used



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.

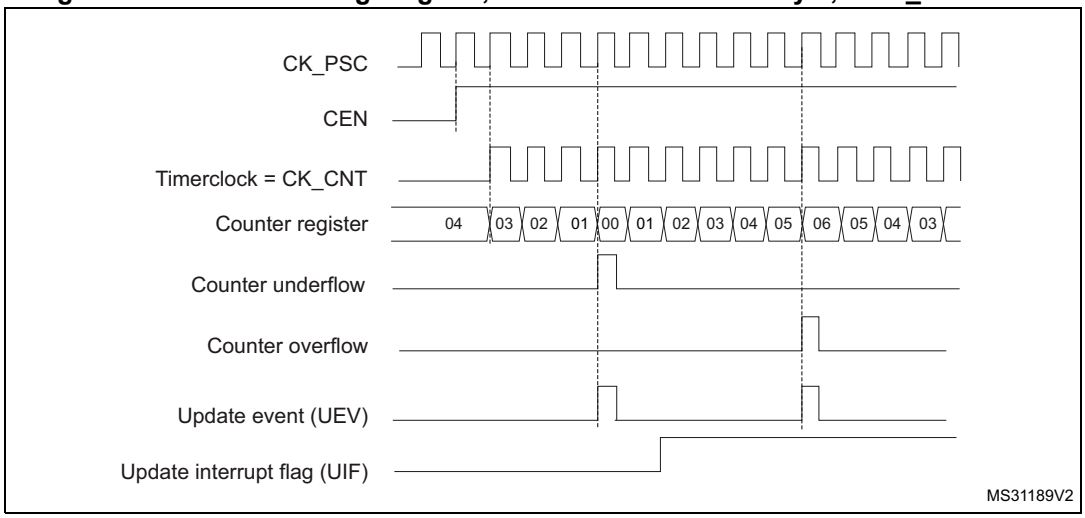
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on 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 100. 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 17.4: TIM1 and TIM8 registers*).

Figure 101. Counter timing diagram, internal clock divided by 2

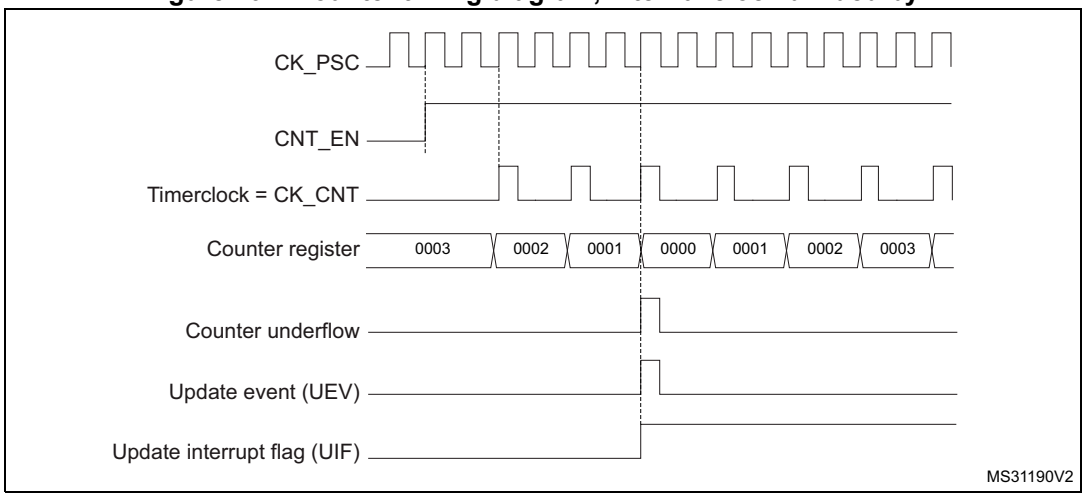
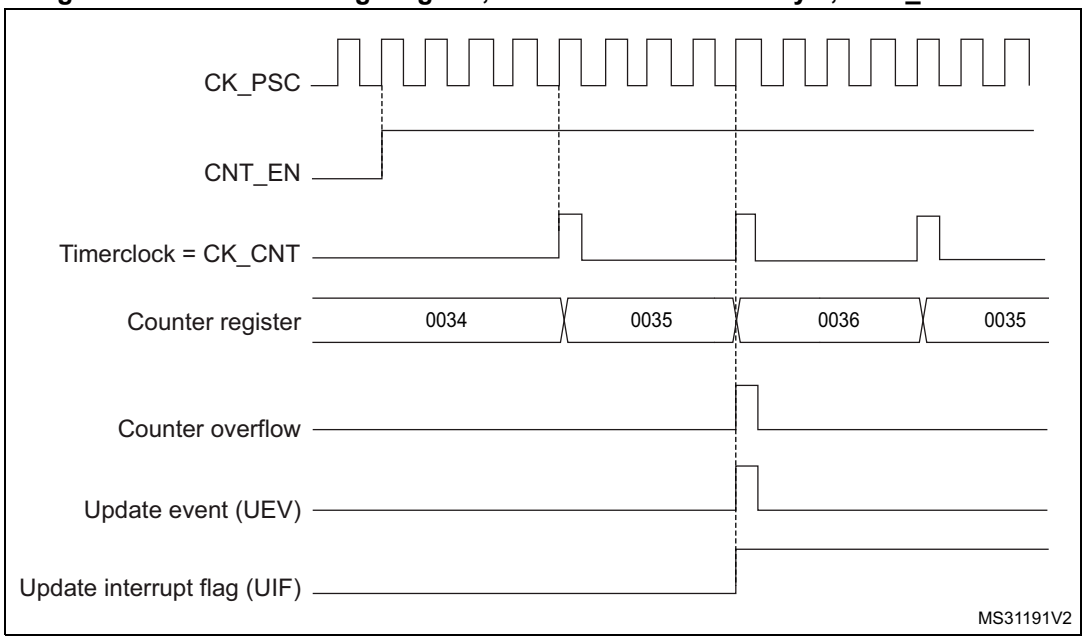


Figure 102. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36



1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 103. Counter timing diagram, internal clock divided by N

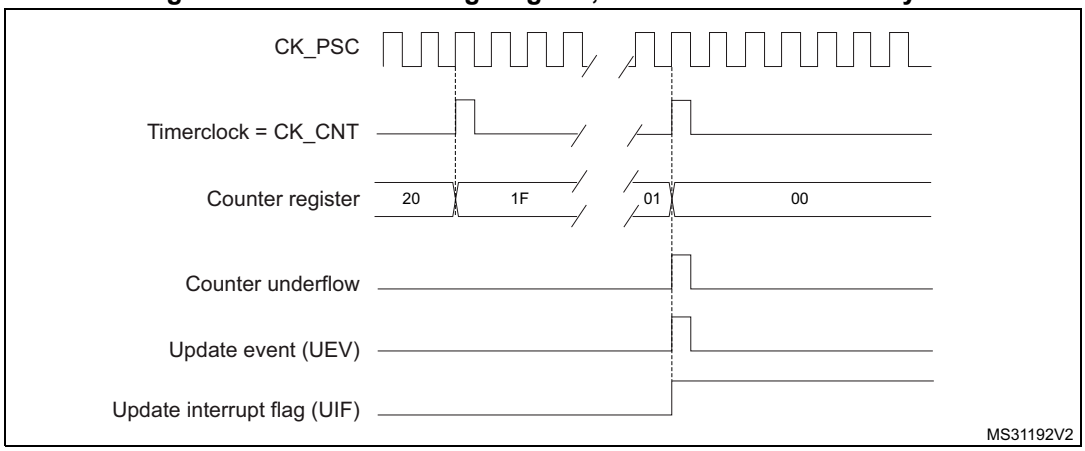


Figure 104. Counter timing diagram, update event with ARPE=1 (counter underflow)

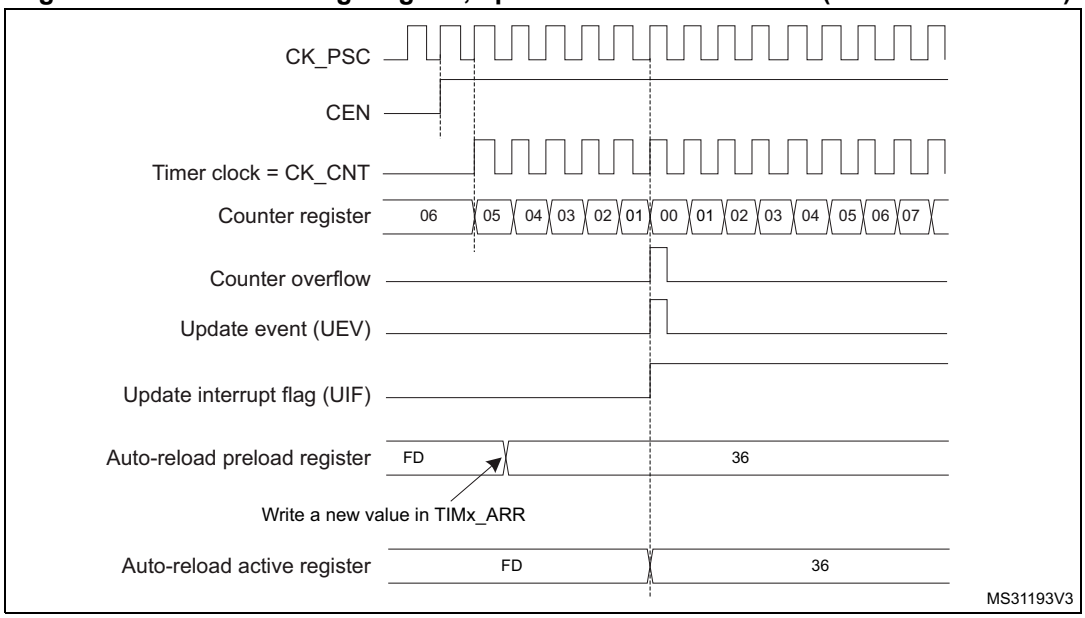
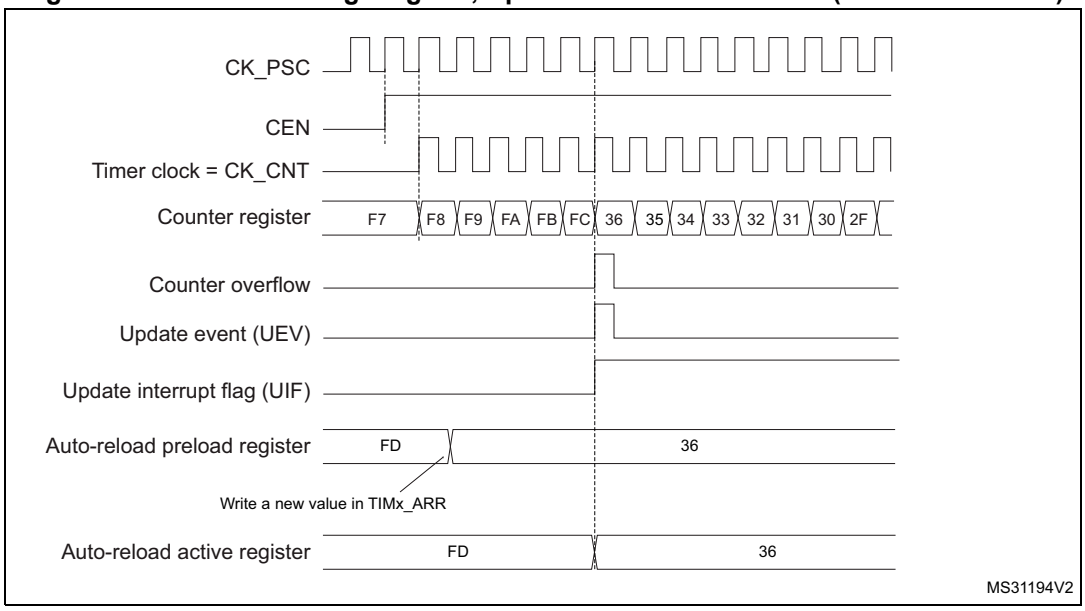


Figure 105. Counter timing diagram, Update event with ARPE=1 (counter overflow)



17.3.3 Repetition counter

Section 17.3.1: 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.

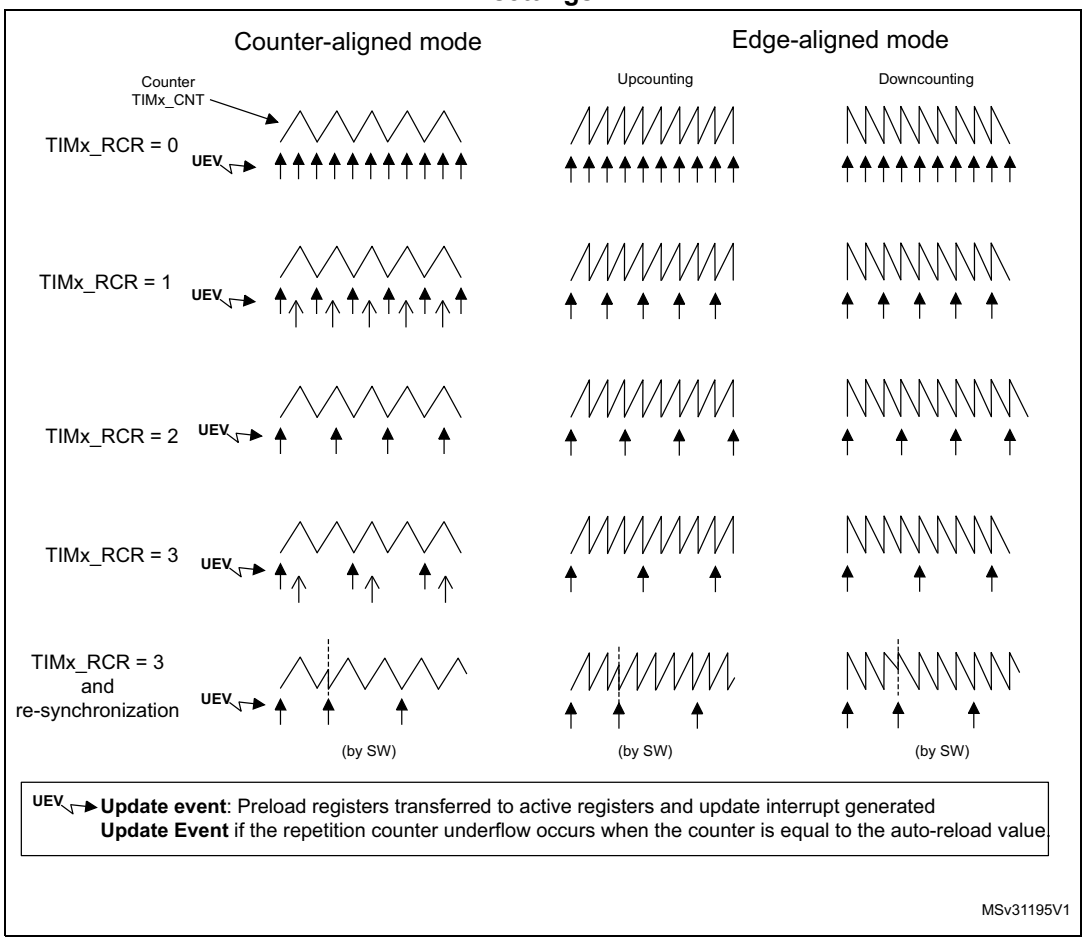
The repetition counter is decremented:

- 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 128 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, 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 106*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

In center-aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was started. If the RCR was written before starting the counter, the UEV occurs on the overflow. If the RCR was written after starting the counter, the UEV occurs on the underflow. For example for RCR = 3, the UEV is generated on each 4th overflow or underflow event depending on when RCR was written.

Figure 106. Update rate examples depending on mode and TIMx_RCR register settings



17.3.4 Clock selection

The counter clock can be provided by the following clock sources:

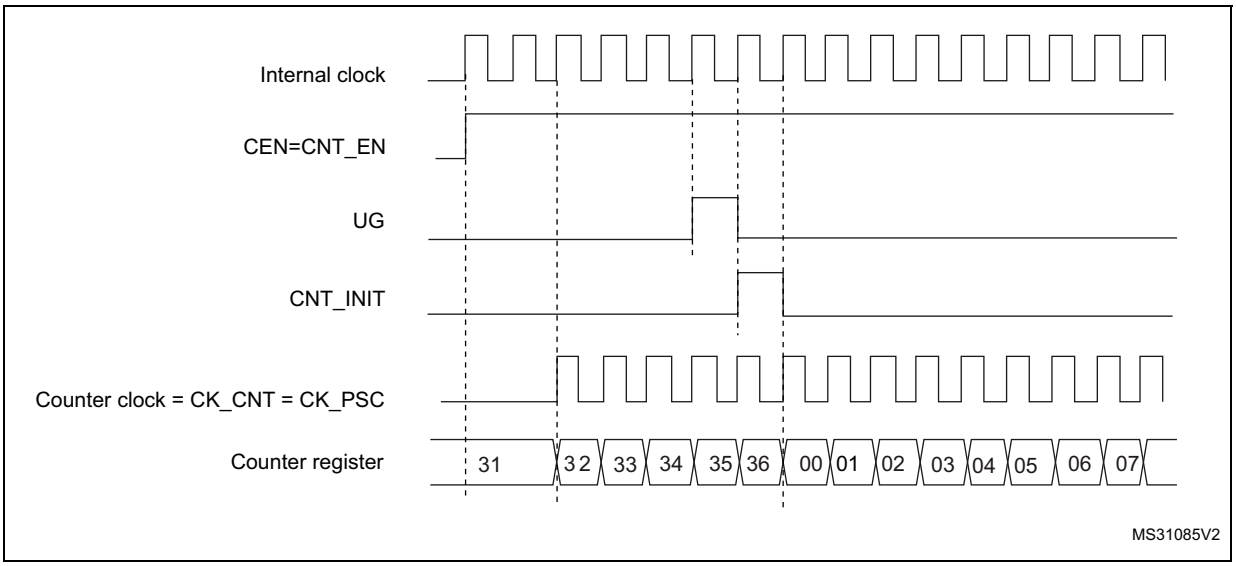
- Internal clock (CK_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, the user can configure Timer 1 to act as a prescaler for Timer 2. Refer to *Using one timer as prescaler for another timer* for more details.

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 107 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

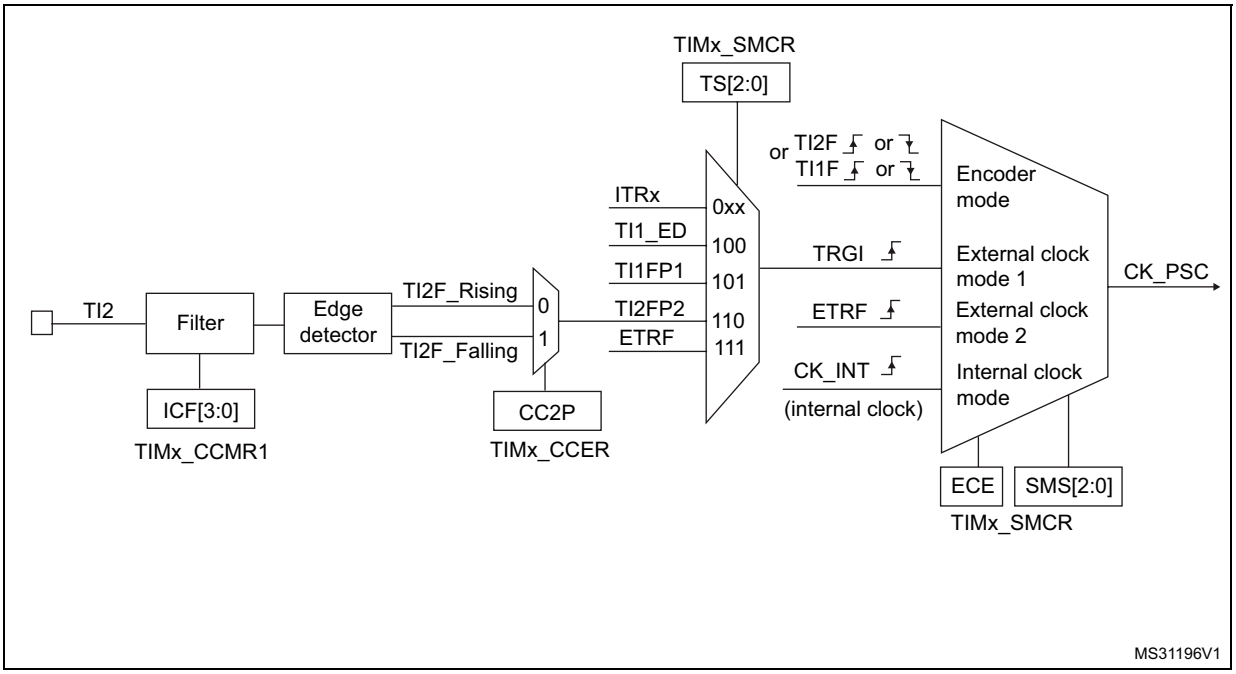
Figure 107. Control circuit in normal mode, internal clock divided by 1



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 108. 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:

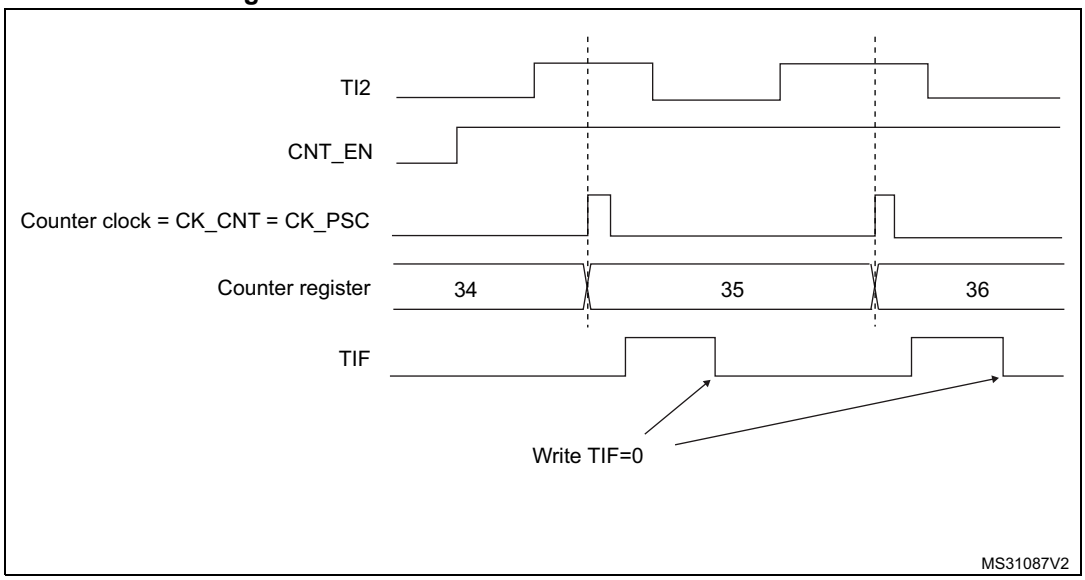
1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 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 109. Control circuit in external clock mode 1



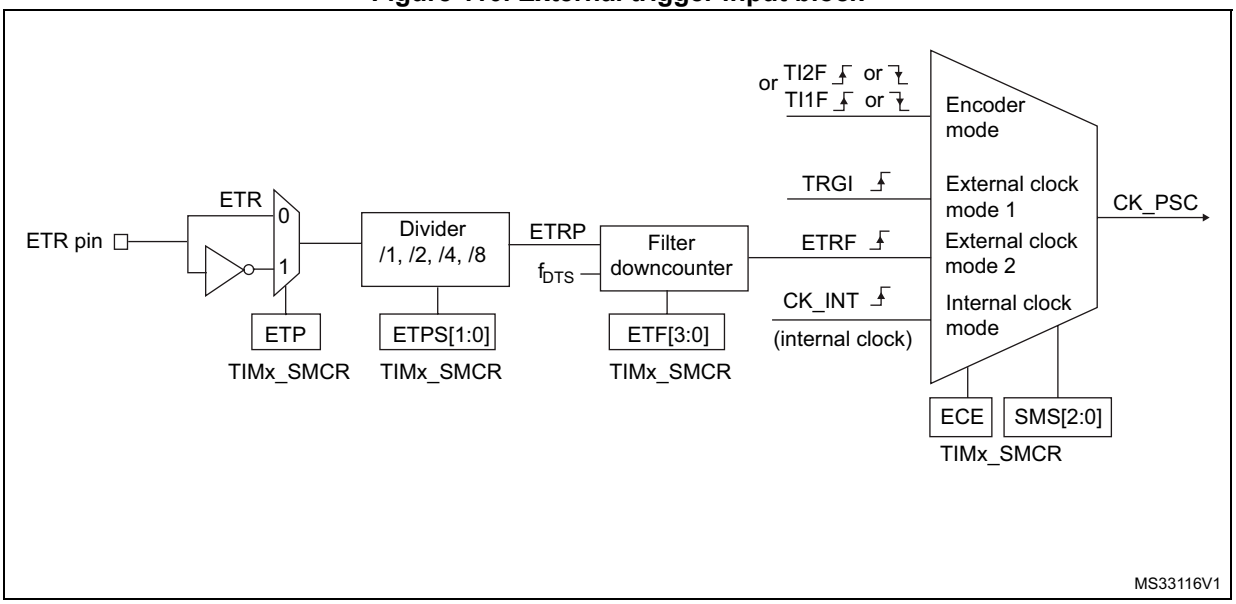
External clock source mode 2

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 110 gives an overview of the external trigger input block.

Figure 110. External trigger input block



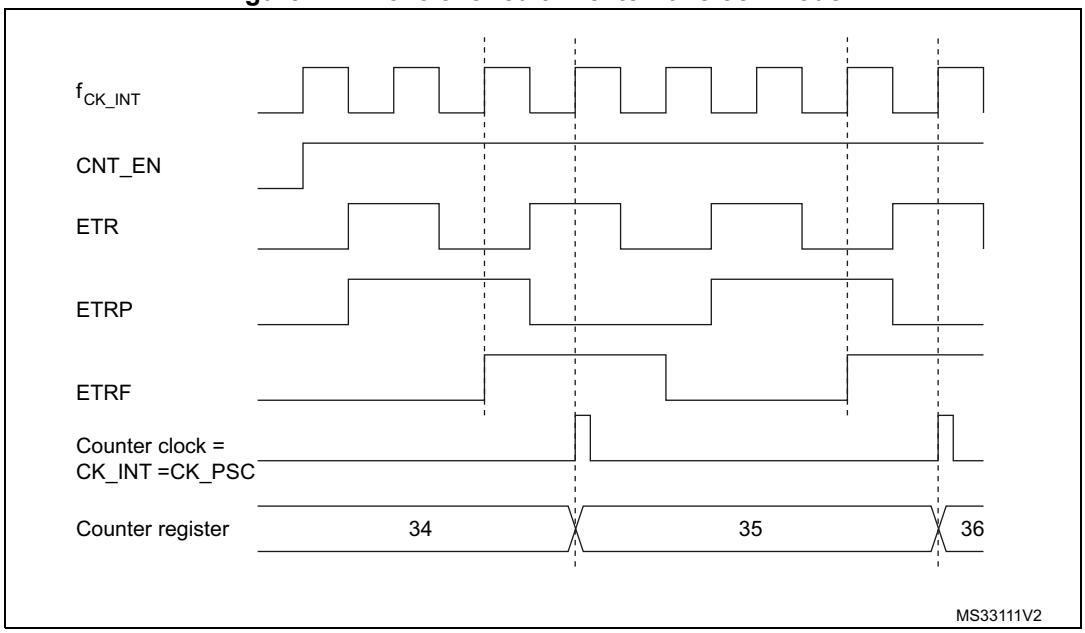
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

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 111. Control circuit in external clock mode 2



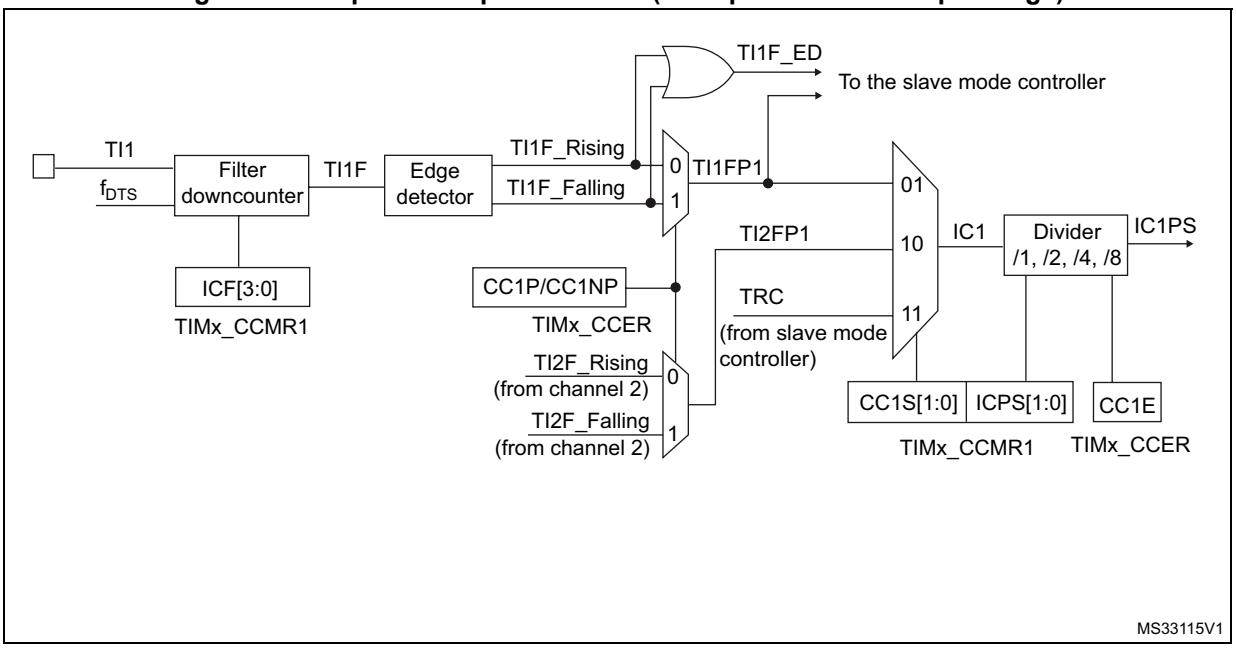
17.3.5 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 112 to *Figure 115* 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 112. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform that is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 113. Capture/compare channel 1 main circuit

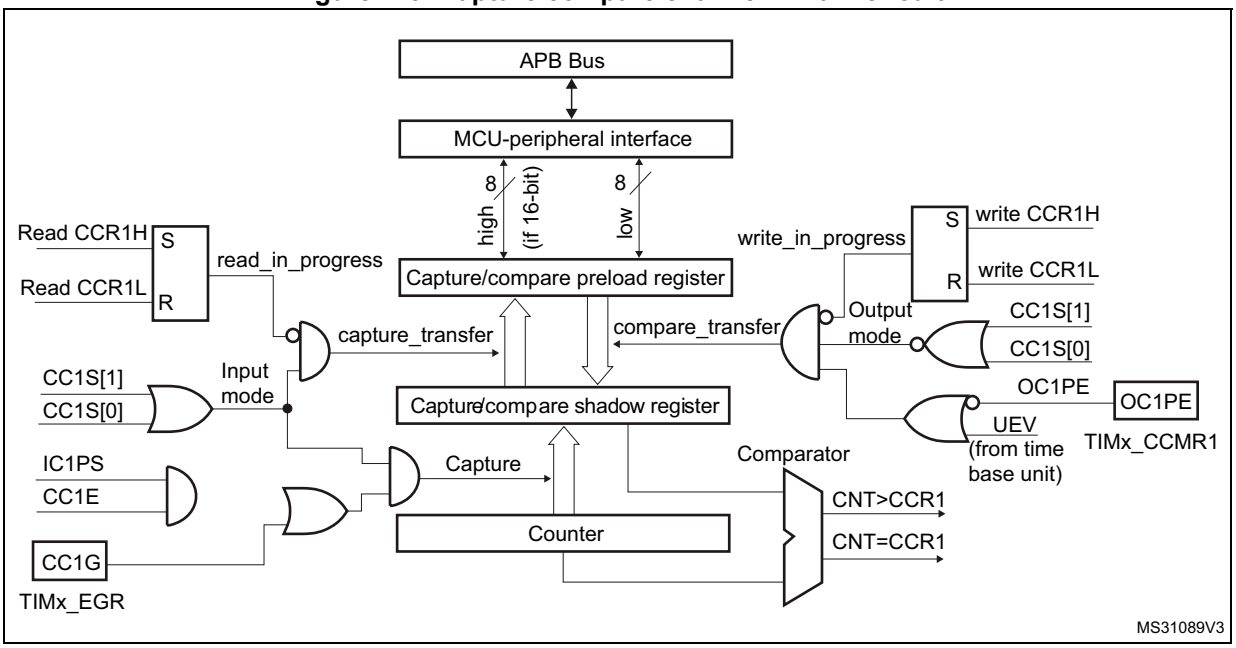


Figure 114. Output stage of capture/compare channel (channel 1 to 3)

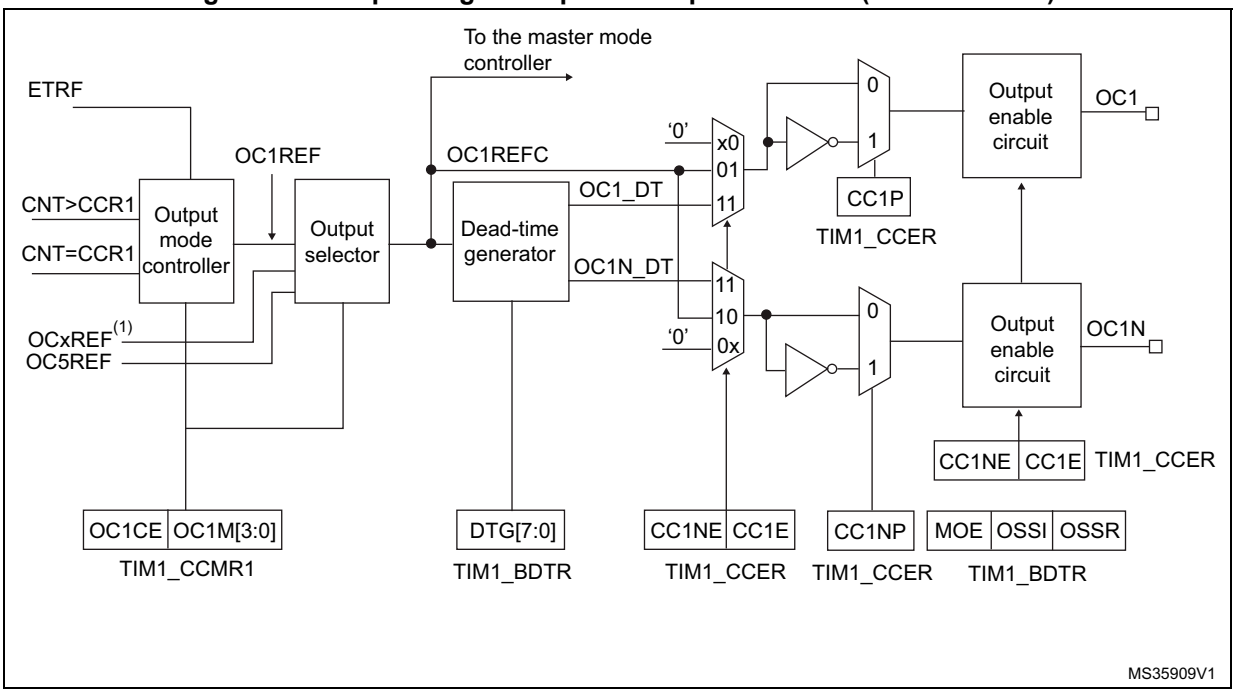
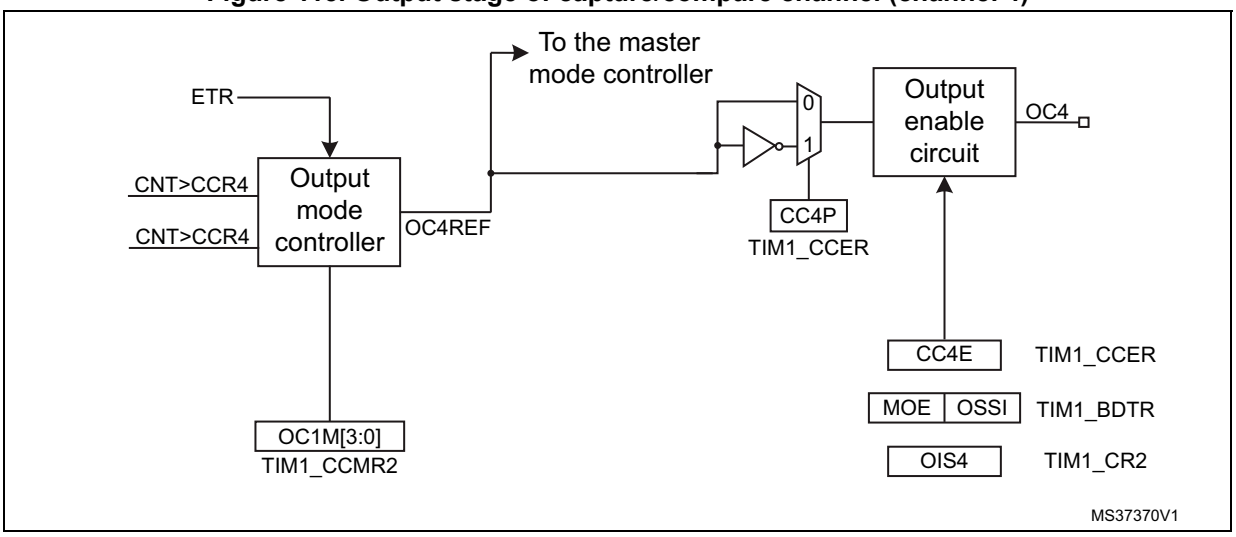


Figure 115. Output stage of capture/compare channel (channel 4)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

17.3.6 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when written to '0'.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
- Program the needed input filter duration with respect to the signal connected to the timer (by programming ICxF bits in the TIMx_CCMRx register if the input is a TIx input). Let's imagine that, when toggling, the input signal is not stable during at must five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
- Select the edge of the active transition on the TI1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

17.3.7 PWM input mode

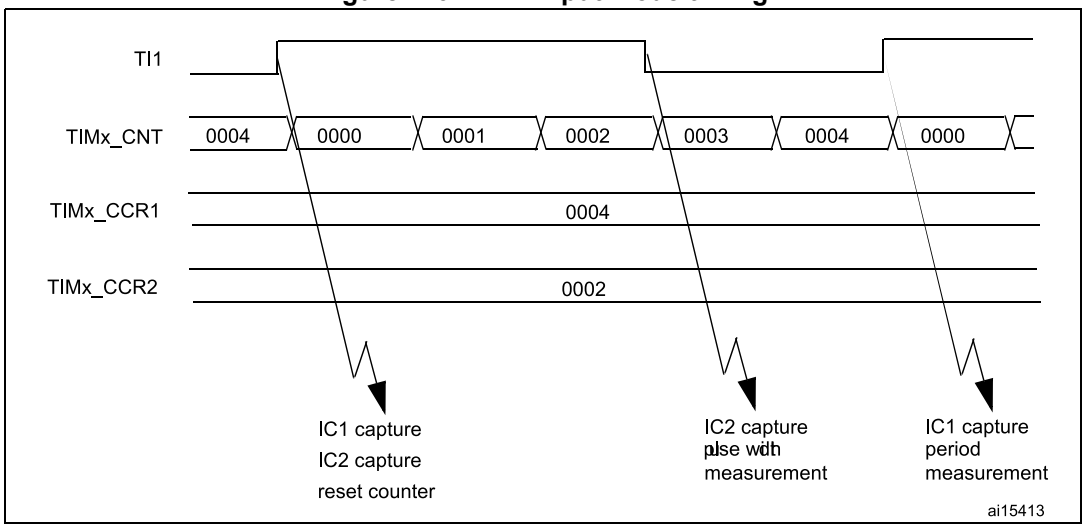
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, 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):

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to '0' (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to '1' (active on falling edge).
- Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 116. PWM input mode timing



17.3.8 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, the user just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is

forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

17.3.9 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

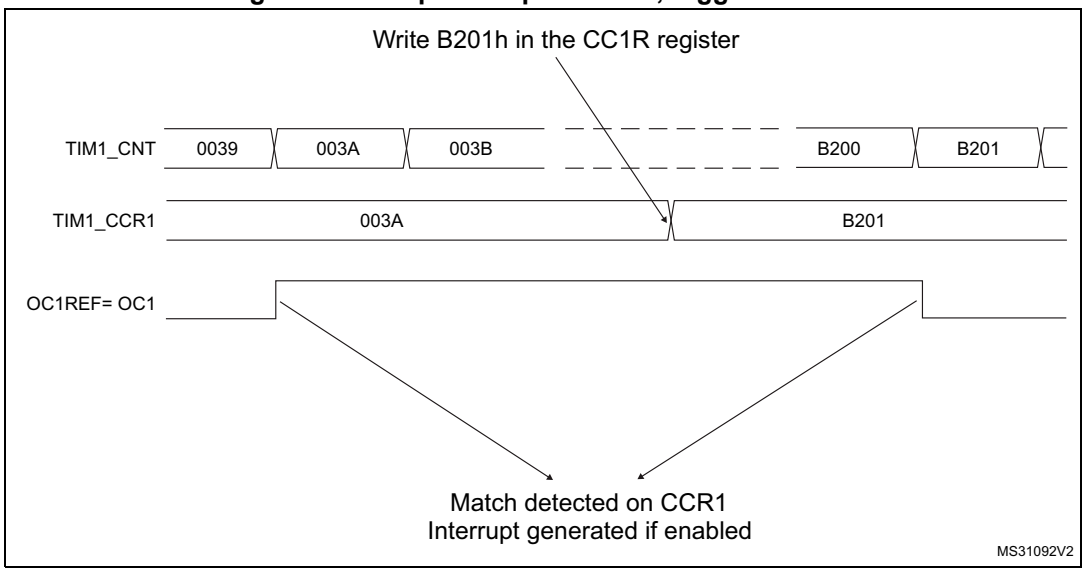
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 117*.

Figure 117. Output compare mode, toggle on OC1.



17.3.10 PWM mode

Pulse Width Modulation mode allows generating a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, the user must 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 ≤TIMx_CNT or TIMx_CNT ≤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

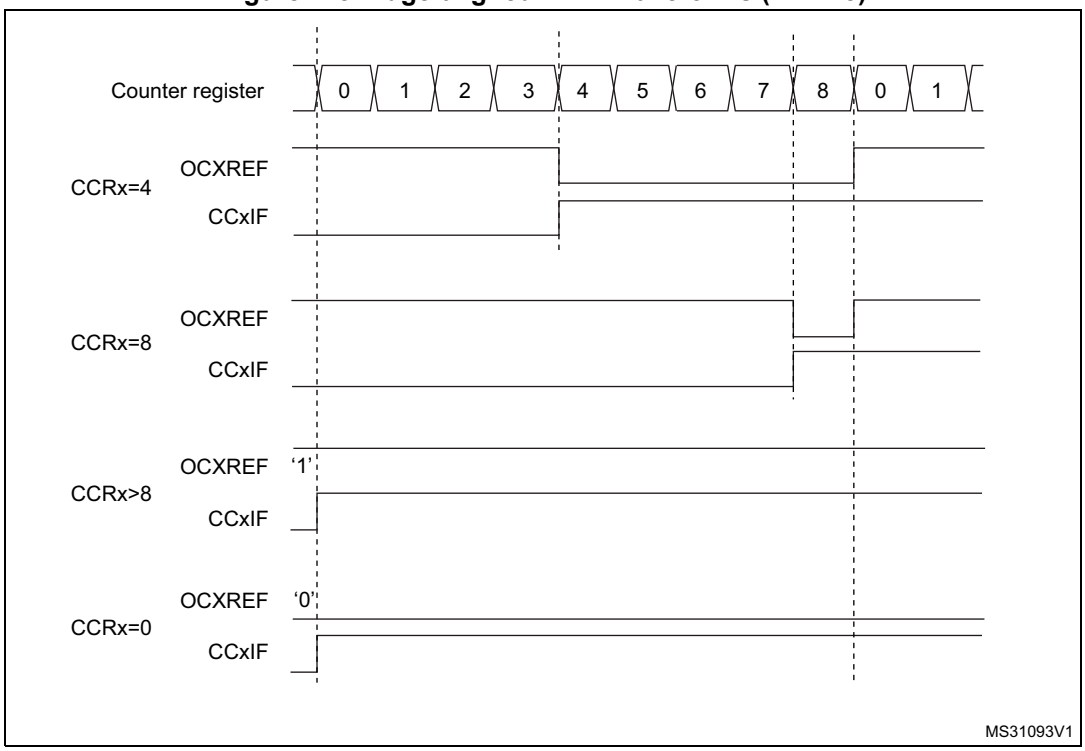
- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode*.

  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 118* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 118. Edge-aligned PWM waveforms (ARR=8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to *Downcounting mode*

  In PWM mode 1, the reference signal OCxRef is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then OCxREF is held at ‘1’. 0% PWM is not possible in this mode.

PWM center-aligned mode

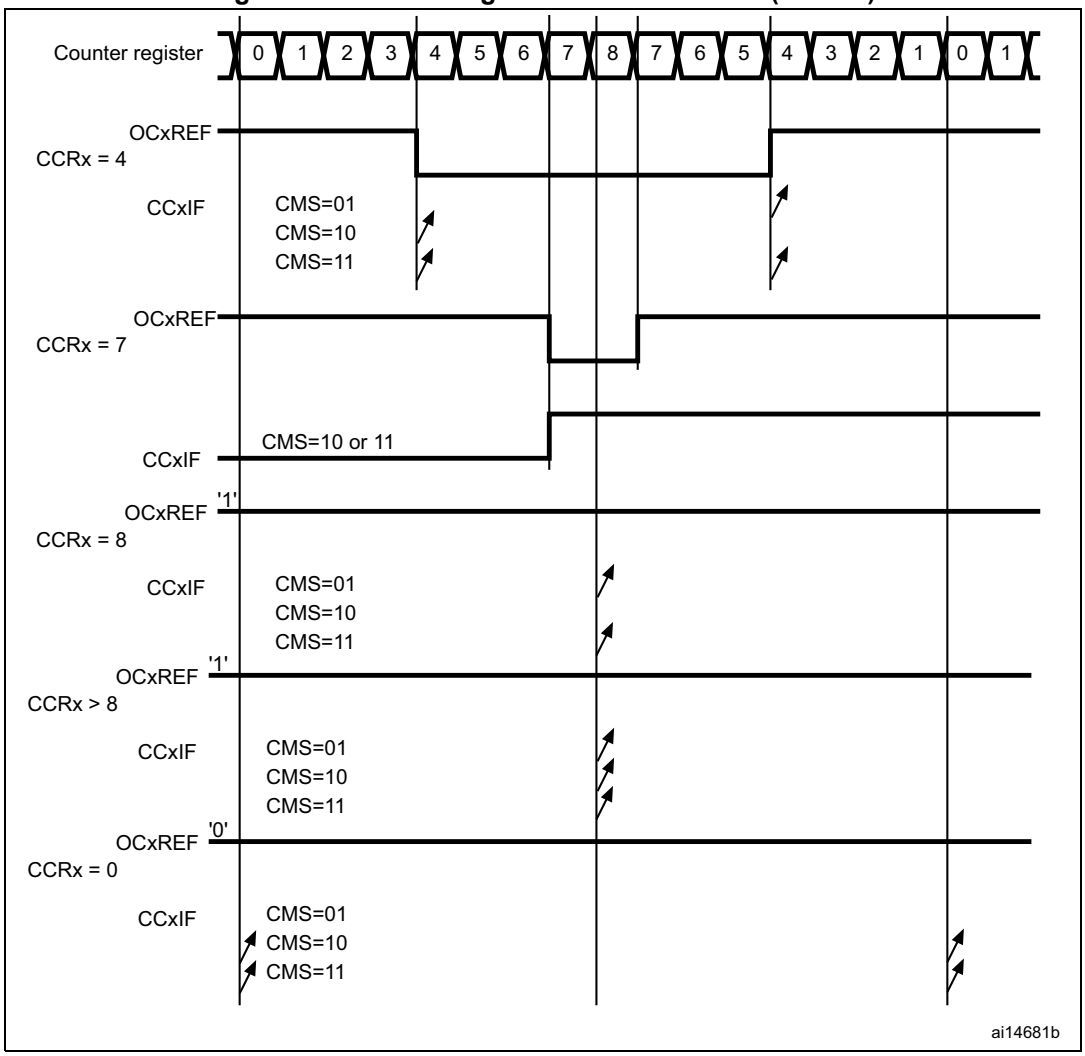
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00’ (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the

TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting)*.

Figure 119 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 119. 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 the user writes a value in the counter greater than the auto-reload value (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it will continue to count up.
  - The direction is updated if the user writes 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.

17.3.11 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1 and 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 it has to be adjust it depending on the devices connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

User 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 95* 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. DTG[7:0] bits of the TIMx_BDTR register are used to control the dead-time generation for all channels. 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 120. Complementary output with dead-time insertion.

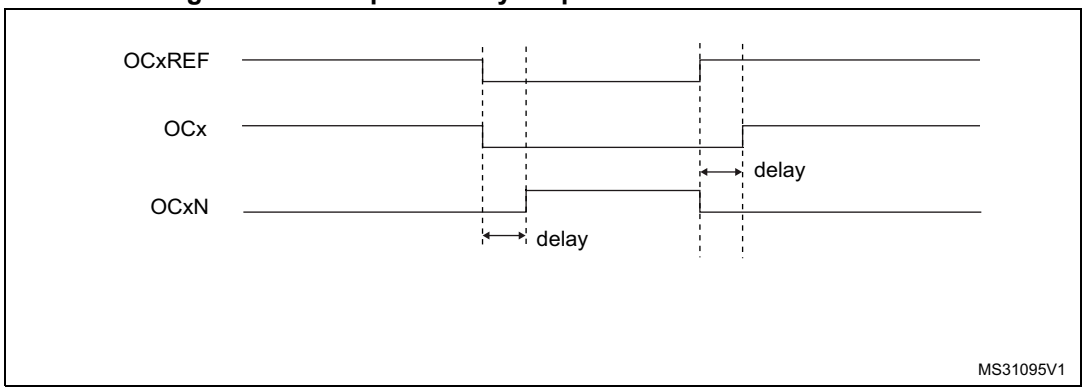


Figure 121. Dead-time waveforms with delay greater than the negative pulse.

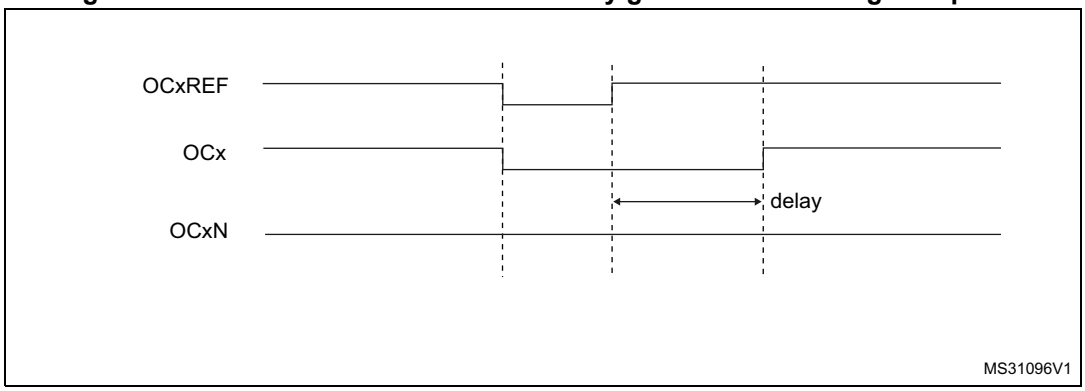
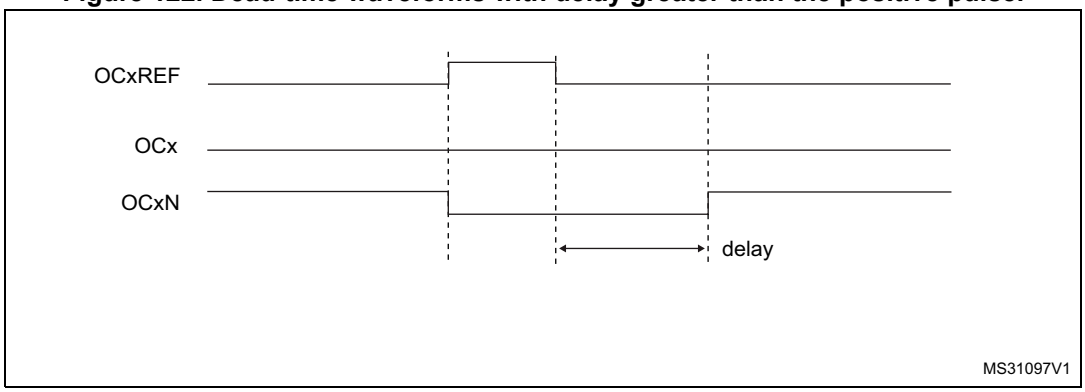


Figure 122. Dead-time waveforms with delay greater than the positive pulse.



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 17.4.18: TIM1 and 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 the user to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other 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.

17.3.12 Using the break function

When using the break function, the output enable signals and inactive levels are modified according to additional control bits (MOE, OSSI and OSSR bits in the TIMx_BDTR register, OISx and OISxN bits in the TIMx_CR2 register). In any case, the OCx and OCxN outputs cannot be set both to active level at a given time. Refer to *Table 95* for more details.

The break source can be either the break input pin or a clock failure event, generated by the Clock Security System (CSS), from the Reset Clock Controller. For further information on the Clock Security System, refer to *Section 7.2.7: Clock security system (CSS)*.

When exiting from reset, the break circuit is disabled and the MOE bit is low. User can enable the break function by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is written to 1 whereas it was low, a delay (dummy instruction) must be inserted before reading it correctly. This is because the user writes an asynchronous signal, but reads a synchronous signal.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or in reset state (selected by the OSSI 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 OSSI=0 then the timer releases the enable output else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their

active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck_tim clock cycles).

– If OSSI=0 then the timer releases the enable outputs else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written to '1' again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: 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.

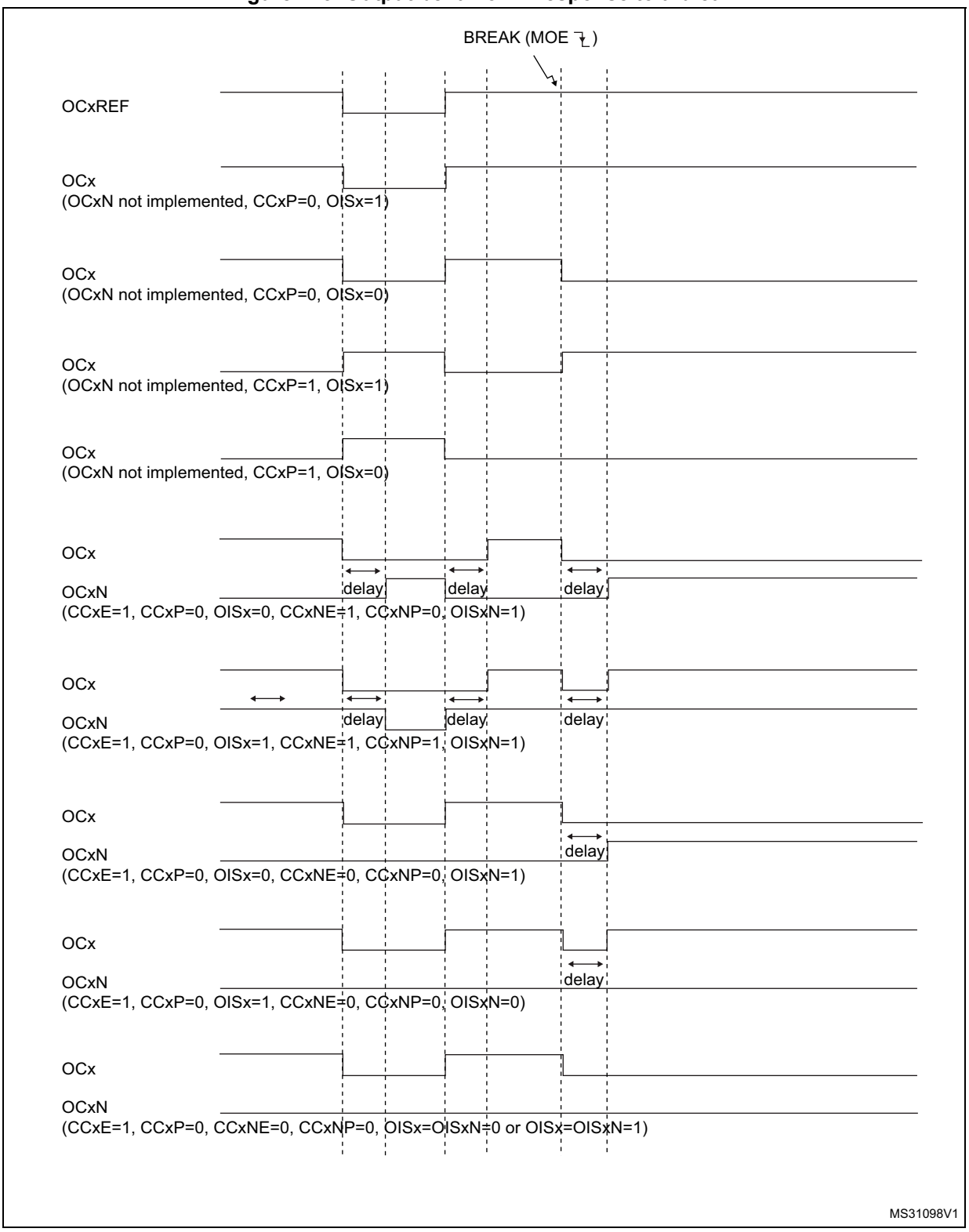
There are two solutions to generate a break:

- By using the BRK input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR register
- By software through the BG bit of the TIMx_EGR 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 freezing the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The user can choose from three levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 17.4.18: TIM1 and TIM8 break and dead-time register (TIMx_BDTR)*. The LOCK bits can be written only once after an MCU reset.

Figure 123 shows an example of behavior of the outputs in response to a break.

Figure 123. Output behavior in response to a break.



17.3.13 Clearing the OCxREF signal on an external event

The OCxREF signal for a given channel can be driven Low by applying a High level to the ETRF input (OCxCE enable bit of the corresponding TIMx_CCMRx register set to ‘1’). The OCxREF signal remains Low until the next update event, UEV, occurs.

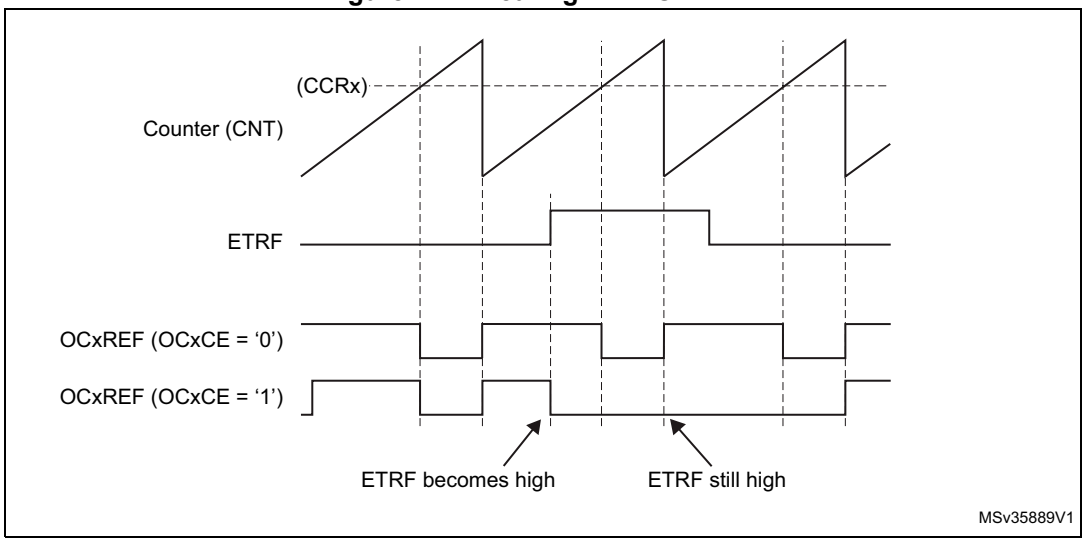
This function can only be used in output compare and PWM modes, and does not work in forced mode.

For example, the ETR signal can be connected to the output of a comparator to be used for current handling. In this case, the ETR must be configured as follow:

1. The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the 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 124 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 124. 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.

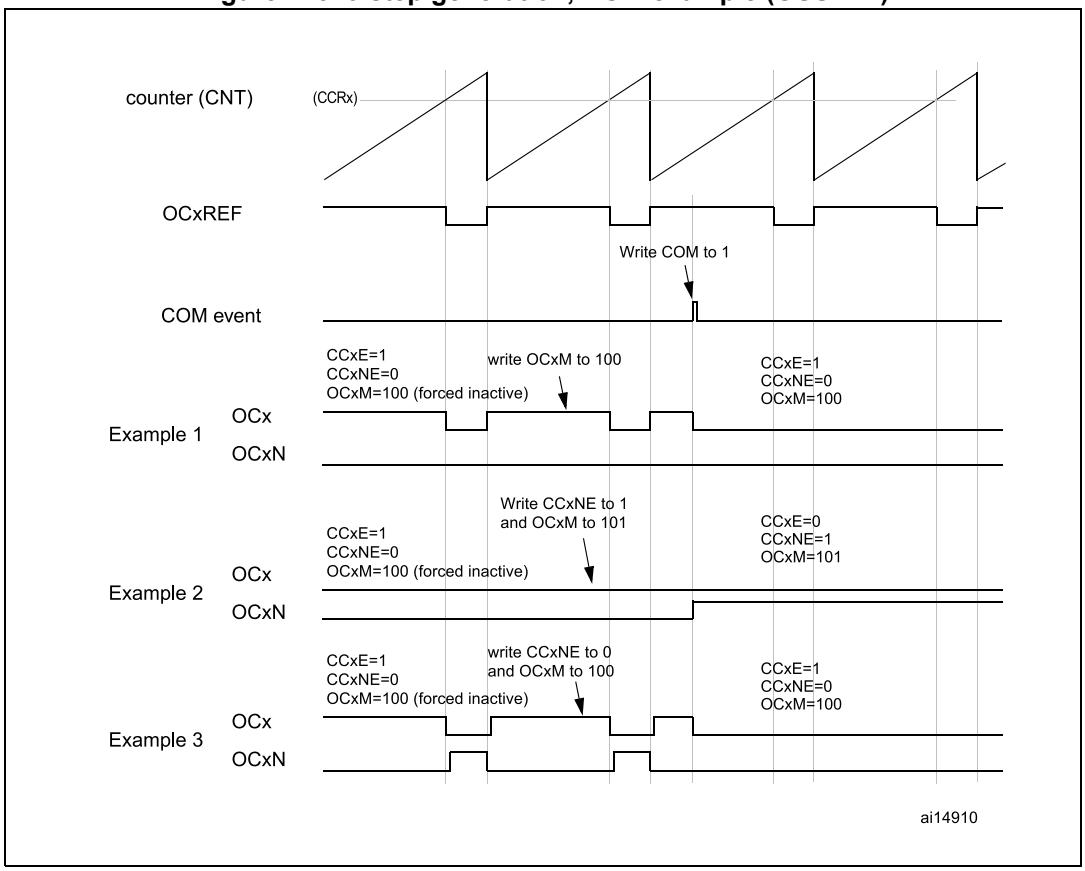
17.3.14 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. The user can thus program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

Figure 125 describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 125. 6-step generation, COM example (OSSR=1)



17.3.15 One-pulse mode

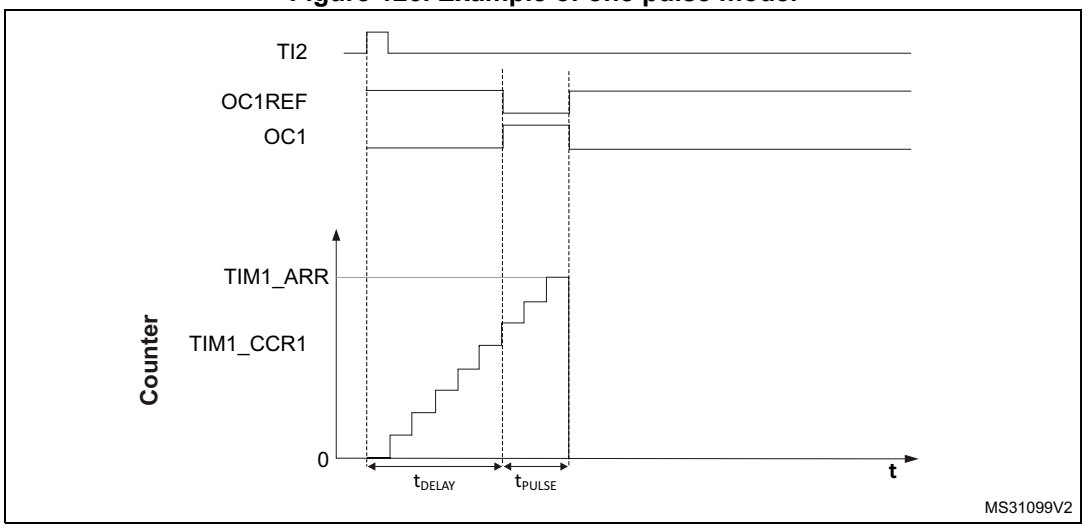
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. 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 ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 126. Example of one pulse mode.



For example the user may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

- Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='110' in the TIMx_SMCR register.
- TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let us say the user wants 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, enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. The user 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 the compare value must be written 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.

The user only wants one pulse (Single mode), so '1’ must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If the user wants to output a waveform with the minimum delay, the OCxFE bit in the TIMx_CCMRx register must be set. 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.

17.3.16 Encoder interface mode

To select Encoder Interface mode write SMS=‘001’ in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS=’010’ if it is counting on TI1 edges only and SMS=’011’ if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the user 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 93*. 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 user must configure TIMx_ARR before starting. in the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the incremental encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. *Table 93* summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 93. 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 127 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S='01' (TIMx_CCMR1 register, TI1FP1 mapped on TI1).
- CC2S='01' (TIMx_CCMR2 register, TI1FP2 mapped on TI2).
- CC1P='0', CC1NP='0', and IC1F = '0000' (TIMx_CCER register, TI1FP1 non-inverted, TI1FP1=TI1).
- CC2P='0', CC2NP='0', and IC2F = '0000' (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 127. Example of counter operation in encoder interface mode.

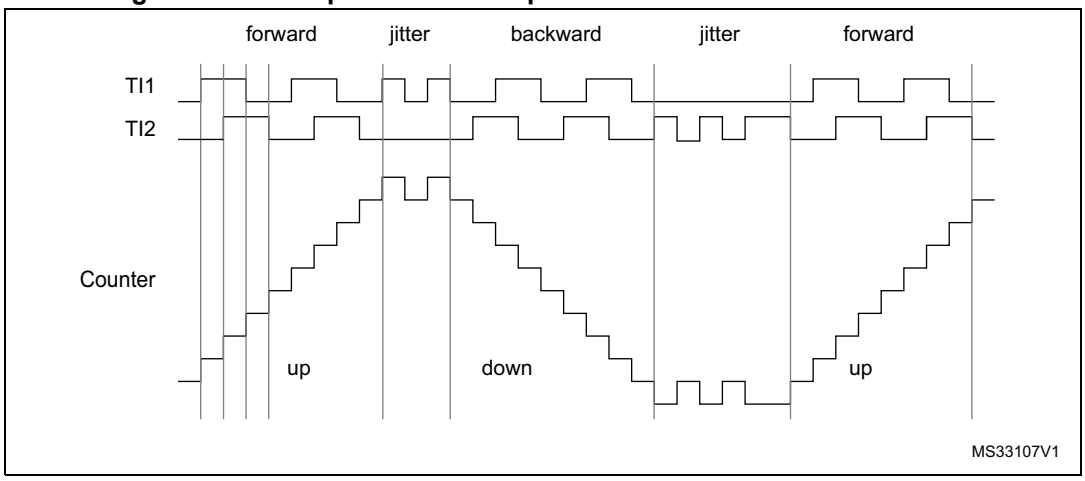
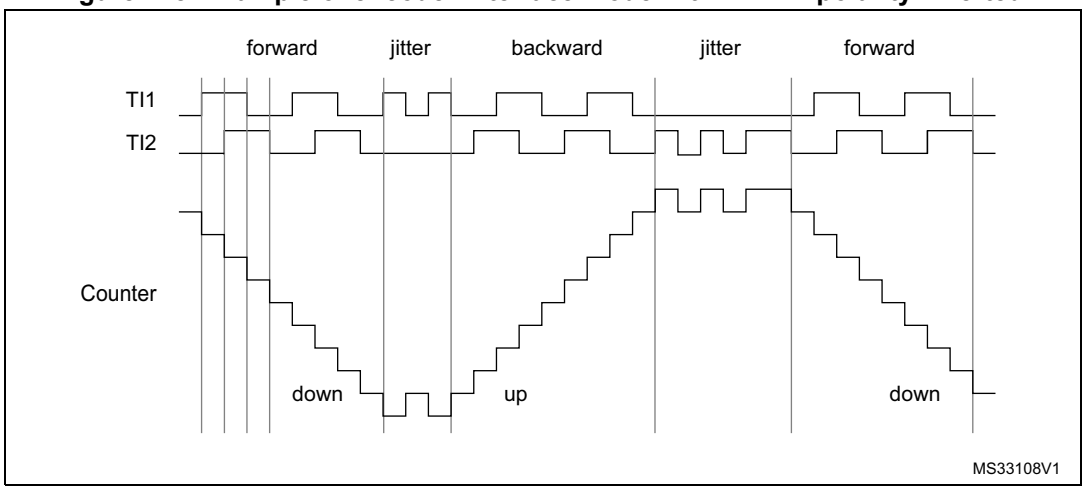


Figure 128 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

Figure 128. 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.The user 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. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request generated by a real-time clock.

17.3.17 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture. An example of this feature used to interface Hall sensors is given in *Section 17.3.18*.

17.3.18 Interfacing with Hall sensors

This is done using the advanced-control timers (TIM1 or TIM8) to generate PWM signals to drive the motor and another timer TIMx (TIM2, TIM3, TIM4 or TIM5) referred to as “interfacing timer” in *Figure 129*. The “interfacing timer” captures the 3 timer input pins (TIMx_CH1, TIMx_CH2, and TIMx_CH3) 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 112*). 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.

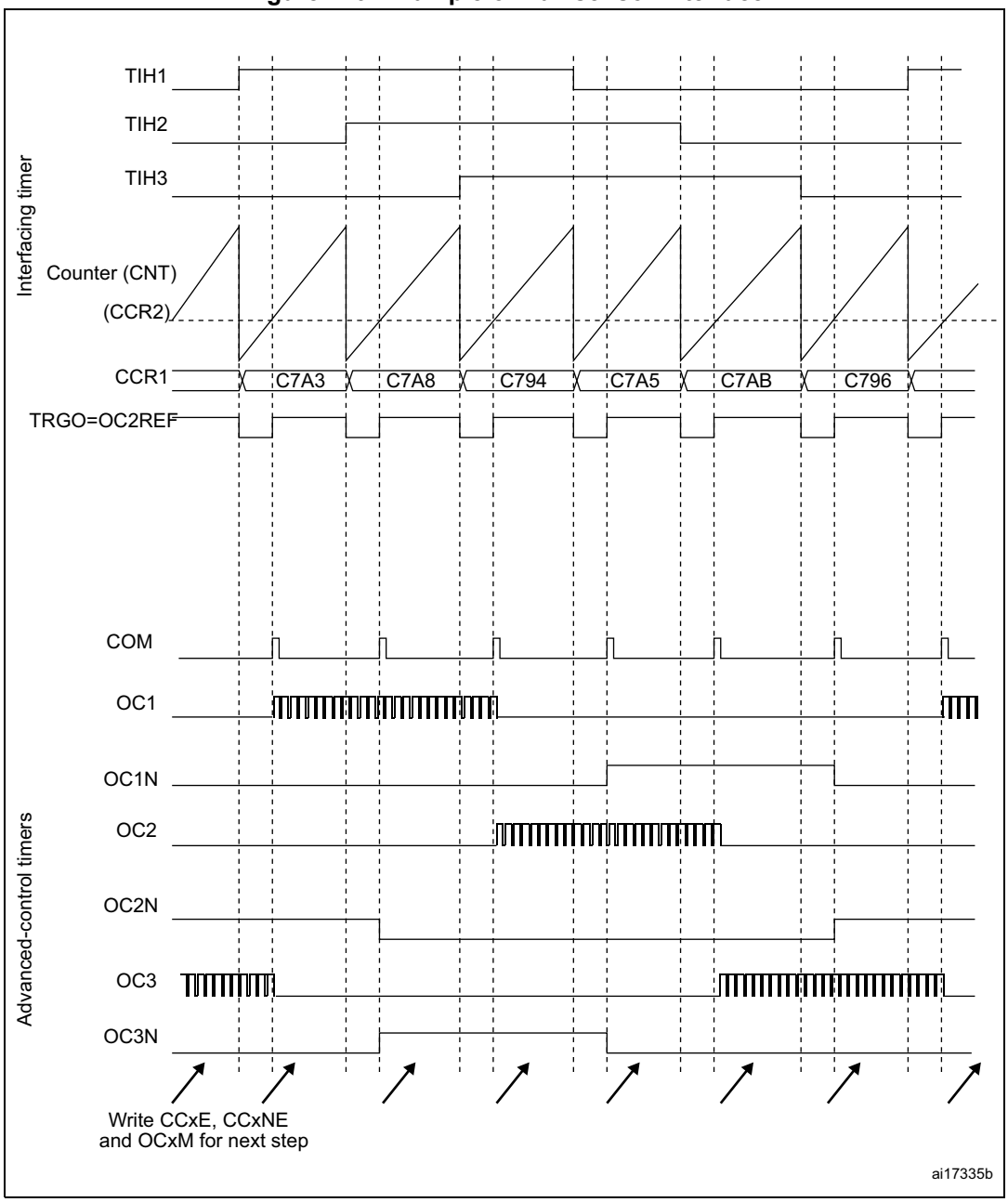
Example: the user wants to change the PWM configuration of the advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx_CR2 register to ‘1’,
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx_CCMR1 register to ‘11’. The user can also program the digital filter if needed,
- Program channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to ‘111’ and the CC2S bits to ‘00’ in the TIMx_CCMR1 register,
- Select OC2REF as trigger output on TRGO: write the MMS bits in the TIMx_CR2 register to ‘101’,

In the advanced-control timer TIM1, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of OC2REF).

Figure 129 describes this example.

Figure 129. Example of Hall sensor interface



17.3.19 TIMx and external trigger synchronization

The TIMx timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

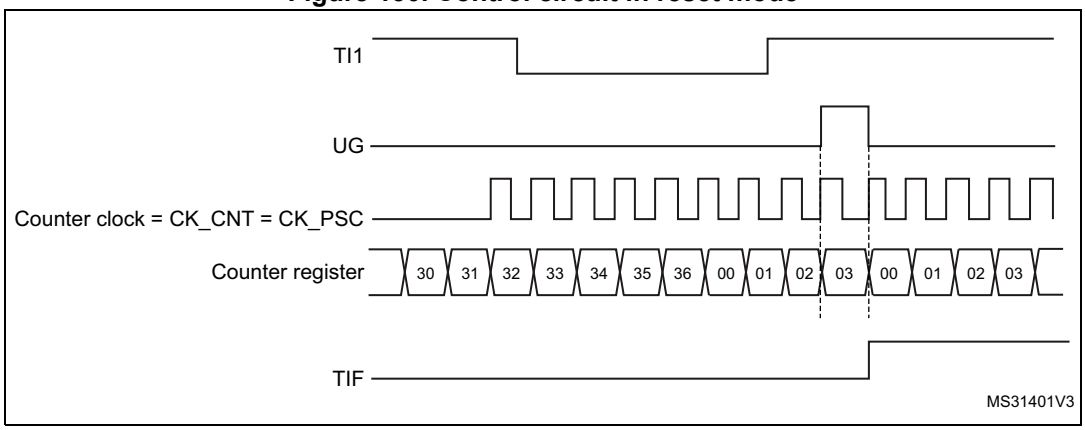
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so there's no 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 130. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

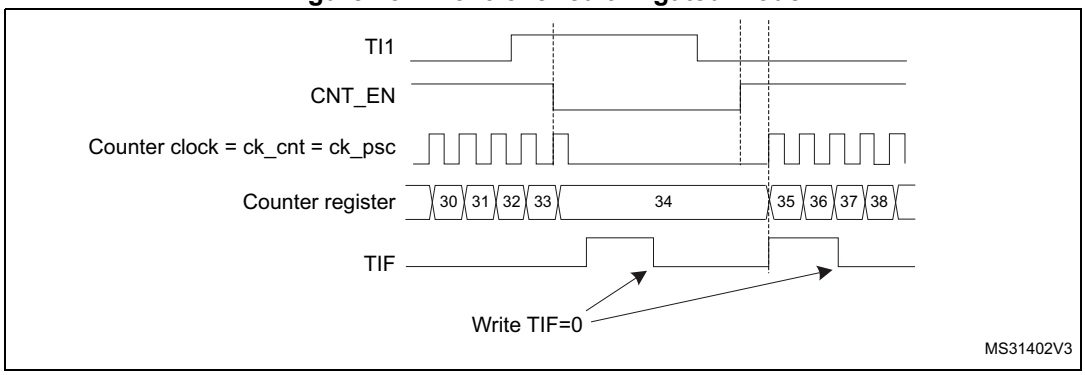
In the following example, the upcounter counts only when TI1 input is low:

- 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 the user does not 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 131. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

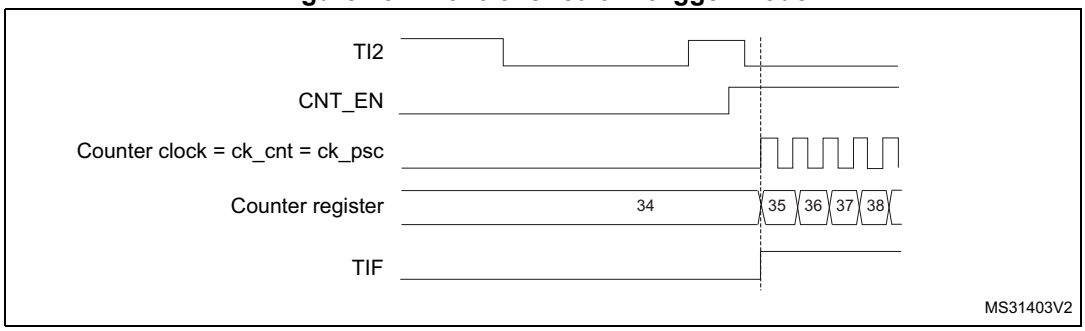
In the following example, the upcounter starts in response to a rising edge on TI2 input:

- 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 there's no 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 132. Control circuit in trigger mode



Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

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

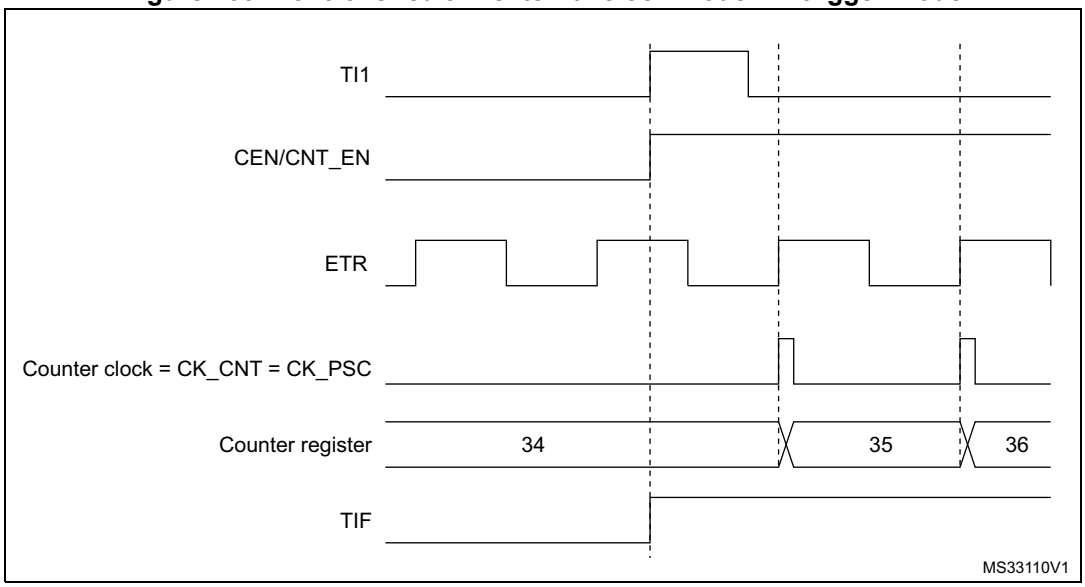
– CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edge only).

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 133. Control circuit in external clock mode 2 + trigger mode



17.3.20 Timer synchronization

The TIM timers are linked together internally for timer synchronization or chaining. Refer to *Section 18.3.15: Timer synchronization* for details.

Note: The clock of the slave timer 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.

17.3.21 Debug mode

When the microcontroller enters debug mode (Cortex®-M4 with FPU 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 38.16.2: Debug support for timers, watchdog, bxCAN and I^2C*.

For safety purposes, when the counter is stopped (DBG_TIMx_STOP = 1 in DBGMCU_APBx_FZ register), 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.

17.4 TIM1 and TIM8 registers

Refer to *Section 1.1* for a list of abbreviations used in register descriptions.

The peripheral registers must be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-word (16 bits) or words (32 bits).

17.4.1 TIM1 and TIM8 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (ETR, TIx),

00: $t_{DTS}=t_{CK\_INT}$
01: $t_{DTS}=2*t_{CK\_INT}$
10: $t_{DTS}=4*t_{CK\_INT}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

17.4.2 TIM1 and TIM8 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

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 and ADC must be enabled prior to receiving 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.

17.4.3 TIM1 and TIM8 slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	Res.	rw	rw	rw

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or $\overline{ETR}$ is used for trigger operations
0: ETR is non-inverted, active at high level or rising edge.
1: ETR is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.
0: External clock mode 2 disabled
1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

*Note: **1:** Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).*

***2:** It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).*

***3:** If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.*

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.
00: Prescaler OFF
01: ETRP frequency divided by 2
10: ETRP frequency divided by 4
11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N 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 94* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SMS:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

000: Slave mode disabled - if CEN = '1' then the prescaler is clocked directly by the internal clock.

001: Encoder mode 1 - Counter counts up/down on TI2FP1 edge depending on TI1FP2 level.

010: Encoder mode 2 - Counter counts up/down on TI1FP2 edge depending on TI2FP1 level.

011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS='100'). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

The clock of the slave timer must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Table 94. TIMx Internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM1	TIM5_TRGO	TIM2_TRGO	TIM3_TRGO	TIM4_TRGO
TIM8	TIM1_TRGO	TIM2_TRGO	TIM4_TRGO	TIM5_TRGO

17.4.4 TIM1 and 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

17.4.5 TIM1 and TIM8 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0	Res.	rc_w0	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

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag
This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred.
1: An active level has been detected on the break input.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag
This flag is set by hardware on COM event (when Capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.
0: No COM event occurred.
1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag
If channel CC1 is configured as output:
This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode)
If channel CC1 is configured as input:
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred
1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
– When CNT is reinitialized by a trigger event (refer to *Section 17.4.3: TIM1 and TIM8 slave mode control register (TIMx_SMCR)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

17.4.6 TIM1 and TIM8 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
								w	w	w	w	w	w	w	w

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

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/Compare control update generation
This bit can be set by software, it is automatically cleared by hardware
0: No action
1: When CCPC bit is set, it allows to update CCxE, CCxNE and OCxM bits
Note: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/Compare 4 generation
refer to CC1G description

Bit 3 **CC3G**: Capture/Compare 3 generation
refer to CC1G description

Bit 2 **CC2G**: Capture/Compare 2 generation
refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

17.4.7 TIM1 and TIM8 capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So the user must take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bit 15 **OC2CE:** Output Compare 2 clear enable

Bits 14:12 **OC2M[2:0]**: Output Compare 2 mode

Bit 11 **OC2PE**: Output Compare 2 preload enable

Bit 10 **OC2FE**: Output Compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7 **OC1CE:** Output Compare 1 clear enable

OC1CE: Output Compare 1 Clear Enable
0: OC1Ref is not affected by the ETRF Input
1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 6:4 **OC1M**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.
000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).
001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.
100: Force inactive level - OC1REF is forced low.
101: Force active level - OC1REF is forced high.
110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0') as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF='1').
111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

***2:** In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.*

3: On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

***2:** The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

Input capture mode

Bits 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).

17.4.8 TIM1 and TIM8 capture/compare mode register 2 (TIMx_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE.	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bit 15 OC4CE: Output compare 4 clear enable

Bits 14:12 **OC4M**: Output compare 4 mode

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable

Bits 6:4 **OC3M**: Output compare 3 mode

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

Input capture mode

Bits 15:12 **IC4F**: Input capture 4 filter

Bits 11:10 **IC4PSC**: Input capture 4 prescaler

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bits 7:4 **IC3F**: Input capture 3 filter

Bits 3:2 **IC3PSC**: Input capture 3 prescaler

Bits 1:0 **CC3S**: Capture/compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

17.4.9 TIM1 and TIM8 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/Compare 4 output polarity
refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable
refer to CC1E description

Bit 11 **CC3NP**: Capture/Compare 3 complementary output polarity
refer to CC1NP description

Bit 10 **CC3NE**: Capture/Compare 3 complementary output enable
refer to CC1NE description

Bit 9 **CC3P**: Capture/Compare 3 output polarity
refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable
refer to CC1E description

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
refer to CC1NP description

Bit 6 **CC2NE**: Capture/Compare 2 complementary output enable
refer to CC1NE description

Bit 5 **CC2P**: Capture/Compare 2 output polarity
refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity
CC1 channel configured as output:
0: OC1N active high.
1: OC1N active low.
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable
0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

CC1 channel configured as output:

0: OC1 active high

1: OC1 active low

CC1 channel configured as input:

CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

00: non-inverted/rising edge

The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

01: inverted/falling edge

The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

10: reserved, do not use this configuration.

11: non-inverted/both edges

The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **CC1E**: Capture/Compare 1 output enable

CC1 channel configured as output:

0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

CC1 channel configured as input:

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.

0: Capture disabled.

1: Capture enabled.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Table 95. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states[1]	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	0	0	0	Output Disabled (not driven by the timer), OCx=0, OCx_EN=0	Output Disabled (not driven by the timer), OCxN=0, OCxN_EN=0
		0	0	1	Output Disabled (not driven by the timer), OCx=0, OCx_EN=0	OCxREF + Polarity OCxN=OCxREF xor CCxNP, OCxN_EN=1
		0	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP, OCx_EN=1	Output Disabled (not driven by the timer) OCxN=0, OCxN_EN=0
		0	1	1	OCREF + Polarity + dead-time OCx_EN=1	Complementary to OCREF (not OCREF) + Polarity + dead-time OCxN_EN=1
		1	0	0	Output Disabled (not driven by the timer) OCx=CCxP, OCx_EN=0	Output Disabled (not driven by the timer) OCxN=CCxNP, OCxN_EN=0
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP, OCx_EN=1	OCxREF + Polarity OCxN=OCxREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP, OCx_EN=1	Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCREF + Polarity + dead-time OCx_EN=1	Complementary to OCREF (not OCREF) + Polarity + dead-time OCxN_EN=1
0	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	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCxN_EN=1 Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (CCxE = CCxNE = 0), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIO registers.

17.4.10 TIM1 and TIM8 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

Bits 15:0 **CNT[15:0]**: Counter value

17.4.11 TIM1 and TIM8 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

17.4.12 TIM1 and 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]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.
Refer to *Section 17.3.1: Time-base unit* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

17.4.13 TIM1 and TIM8 repetition counter register (TIMx_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								REP[7:0]							
								rw	rw	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

– the number of half PWM period in center-aligned mode.

17.4.14 TIM1 and 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/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

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). The TIMx_CCR1 register is read-only and cannot be programmed.

17.4.15 TIM1 and 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/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

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). The TIMx_CCR2 register is read-only and cannot be programmed.

17.4.16 TIM1 and 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/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

Bits 15:0 **CCR3[15:0]**: Capture/Compare value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx_CCR3 register is read-only and cannot be programmed.

17.4.17 TIM1 and 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/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

Bits 15:0 **CCR4[15:0]**: Capture/Compare value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR3 register is read-only and cannot be programmed.

17.4.18 TIM1 and TIM8 break and dead-time register (TIMx_BDTR)

Address offset: 0x44

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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 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.

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: OC and OCN outputs are disabled or forced to idle state.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).

See OC/OCN enable description for more details (*Section 17.4.9: TIM1 and 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 the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity
0: Break input BRK is active low
1: Break input BRK is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable
0: Break inputs (BRK and CSS clock failure event) disabled
1; Break inputs (BRK and CSS clock failure event) enabled
Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode
This bit is used when MOE=1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.
See OC/OCN enable description for more details (*Section 17.4.9: TIM1 and TIM8 capture/compare enable register (TIMx_CCER)*).
0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0).
1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1. Then, OC/OCN enable output signal=1
Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode
This bit is used when MOE=0 on channels configured as outputs.
See OC/OCN enable description for more details (*Section 17.4.9: TIM1 and TIM8 capture/compare enable register (TIMx_CCER)*).
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])xt_{dtg} with $T_{dtg}=2xt_{DTS}$.
DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$.
DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$.
Example if T_{DTS}=125ns (8MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 us to 31750 ns by 250 ns steps,
32 us to 63us by 1 us steps,
64 us to 126 us by 2 us steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

17.4.19 TIM1 and TIM8 DMA control register (TIMx_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DBL[4:0]					Reserved			DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

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 detects a burst transfer when a read or a write access to the TIMx_DMAR register address is performed).
the TIMx_DMAR address)
00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
10001: 18 transfers

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

17.4.20 TIM1 and 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]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

Example of how to use the DMA burst feature

In this example the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

Note: This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

17.4.21 TIM1 and TIM8 register map

TIM1 and TIM8 registers are mapped as 16-bit addressable registers as described in the table below:

Table 96. TIM1 and TIM8 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Reserved																						CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x04	**TIMx_CR2**	Reserved																	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Reserved	CCPC
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x08	**TIMx_SMCR**	Reserved																ETP	ECE	ETP S [1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	**TIMx_DIER**	Reserved																	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**TIMx_SR**	Reserved																			CC4OF	CC3OF	CC2OF	CC1OF	Reserved	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0		0	0	0	0	0	0	0	0
0x14	**TIMx_EGR**	Reserved																								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																									0	0	0	0	0	0	0	0
0x18	**TIMx_CCMR1** Output Compare mode	Reserved																OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	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	Reserved																IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIMx_CCMR2** Output Compare mode	Reserved																O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Input Capture mode	Reserved																IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIMx_CCER**	Reserved																		CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**TIMx_CNT**	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 96. TIM1 and TIM8 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	**TIMx_ARR**	Reserved																ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIMx_RCR**	Reserved																								REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIMx_CCR1**	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIMx_CCR2**	Reserved																CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**TIMx_CCR3**	Reserved																CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**TIMx_CCR4**	Reserved																CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**TIMx_BDTR**	Reserved																MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**TIMx_DCR**	Reserved																			DBL[4:0]					Reserved		DBA[4:0]				
	Reset value																				0	0	0	0	0			0	0	0	0	0
0x4C	**TIMx_DMAR**	DMAB[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 2.3: Memory map* for the register boundary addresses.

18 General-purpose timers (TIM2 to TIM5)

This section applies to the whole STM32F4xx family, unless otherwise specified.

18.1 TIM2 to TIM5 introduction

The general-purpose timers consist of a 16-bit or 32-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

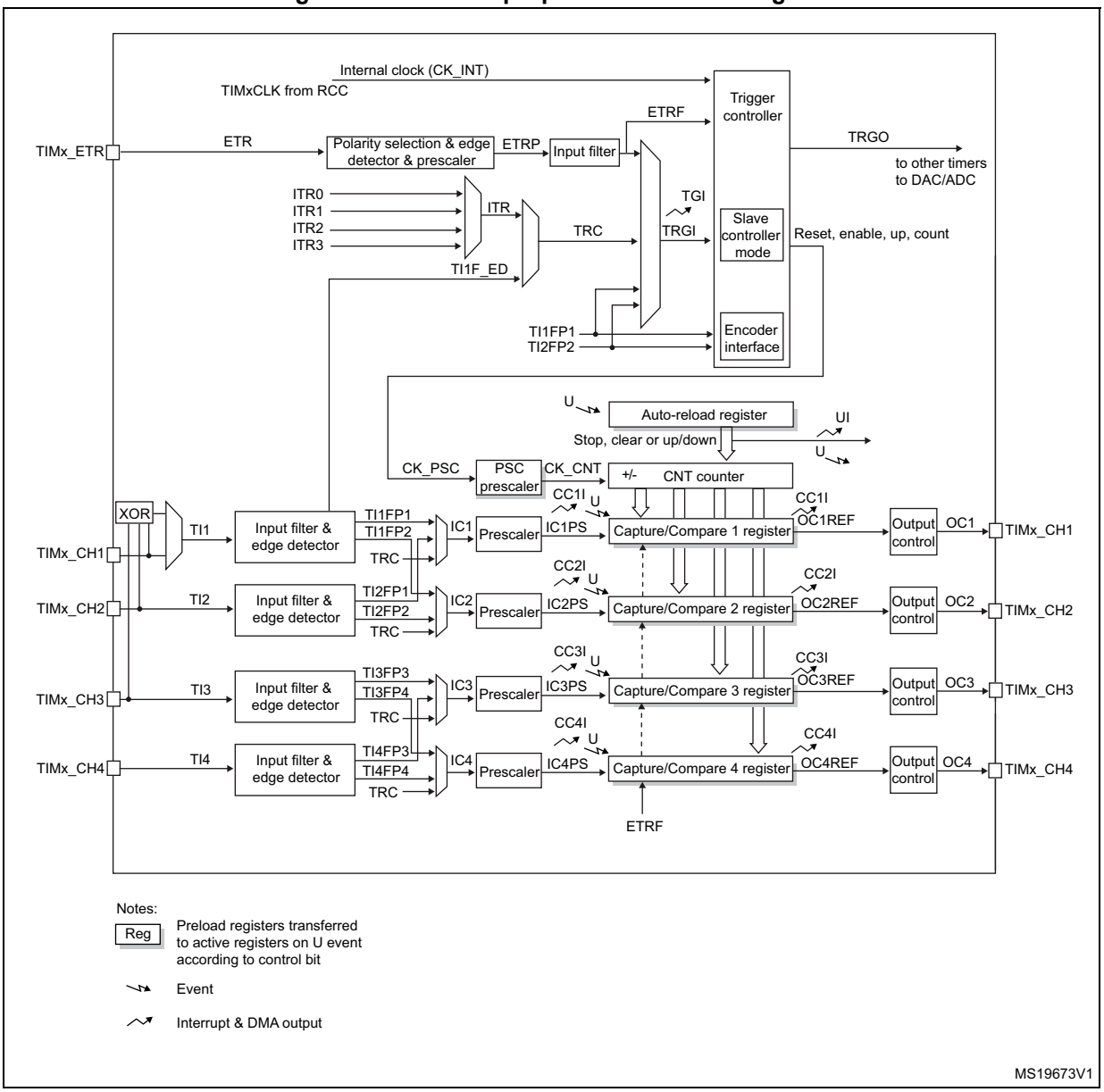
The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 18.3.15*.

18.2 TIM2 to TIM5 main features

General-purpose TIMx timer features include:

- 16-bit (TIM3 and TIM4) or 32-bit (TIM2 and TIM5) up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65536.
- 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 134. General-purpose timer block diagram



18.3 TIM2 to TIM5 functional description

18.3.1 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related auto-reload register. The counter can count up. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC):
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 135 and *Figure 136* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 135. Counter timing diagram with prescaler division change from 1 to 2

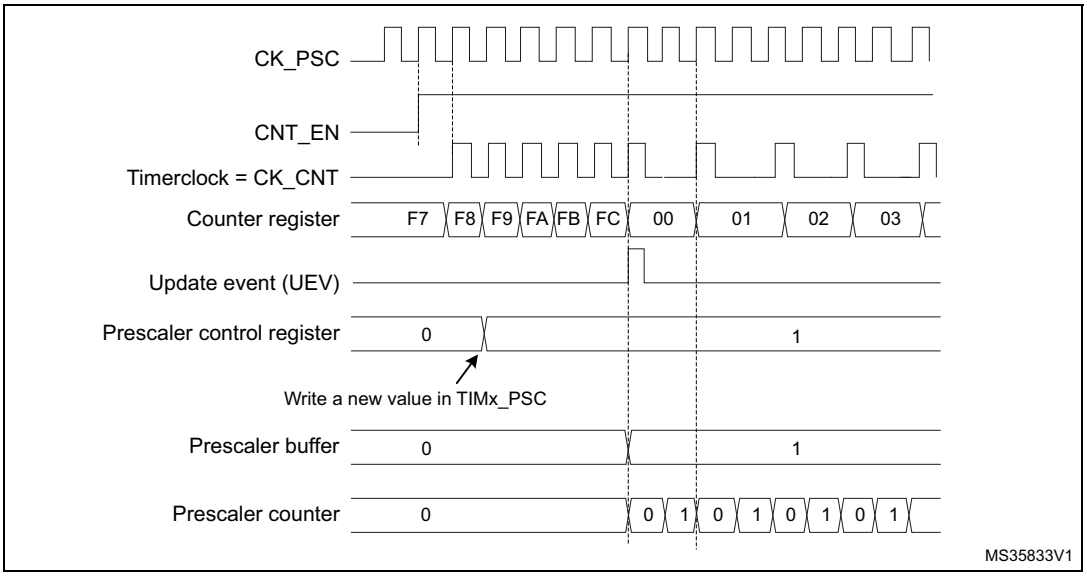
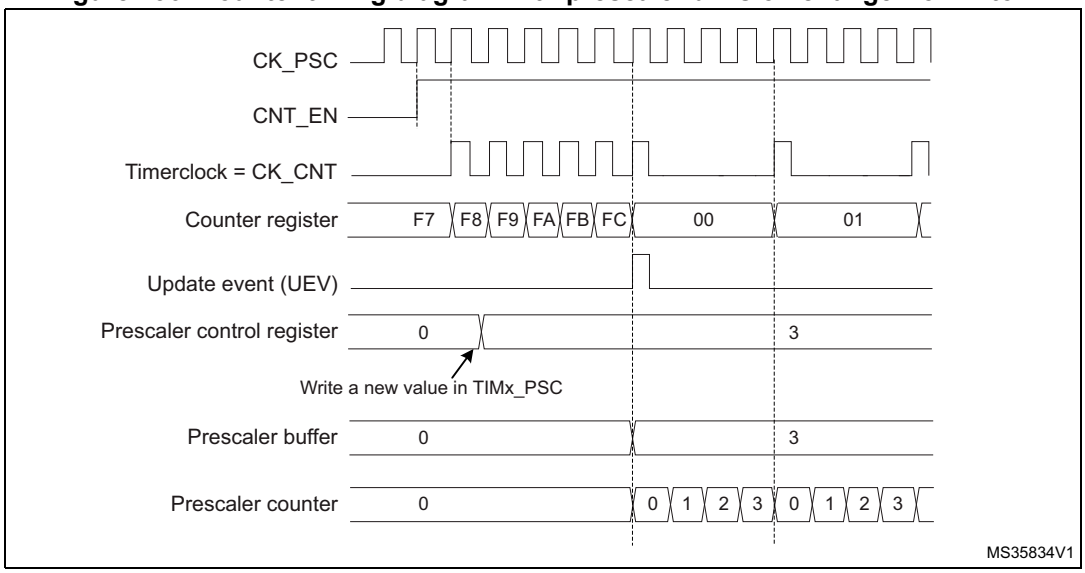


Figure 136. Counter timing diagram with prescaler division change from 1 to 4



18.3.2 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 137. Counter timing diagram, internal clock divided by 1

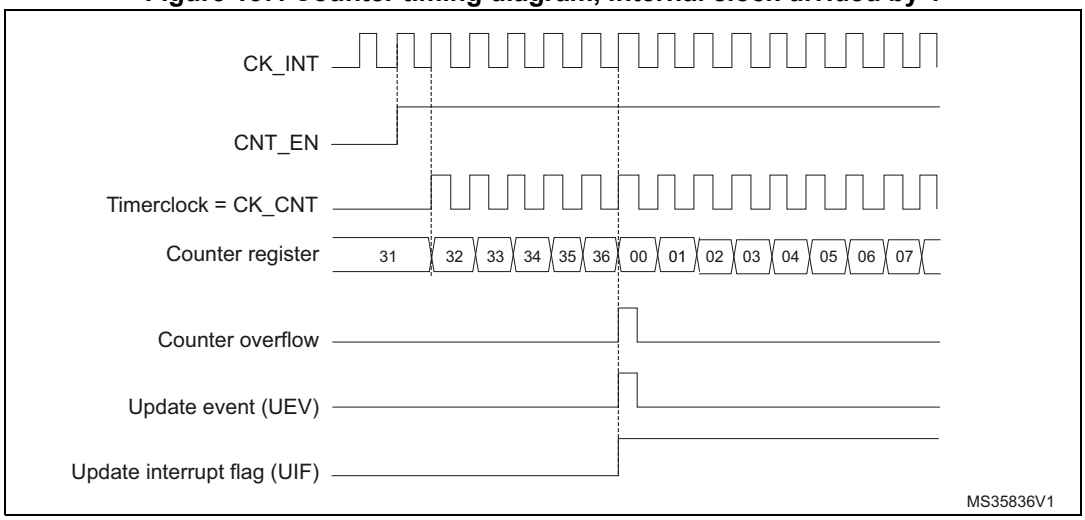


Figure 138. Counter timing diagram, internal clock divided by 2

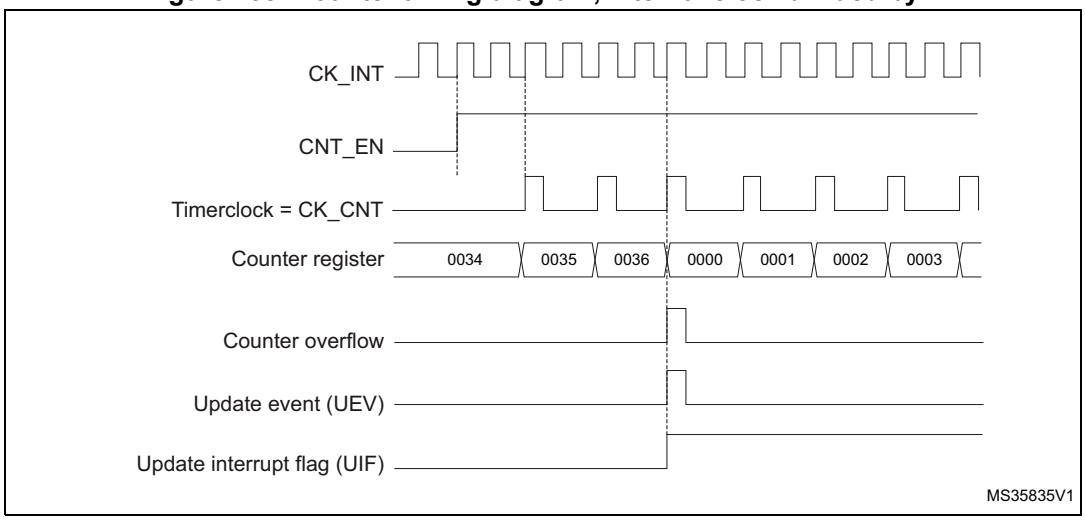


Figure 139. Counter timing diagram, internal clock divided by 4

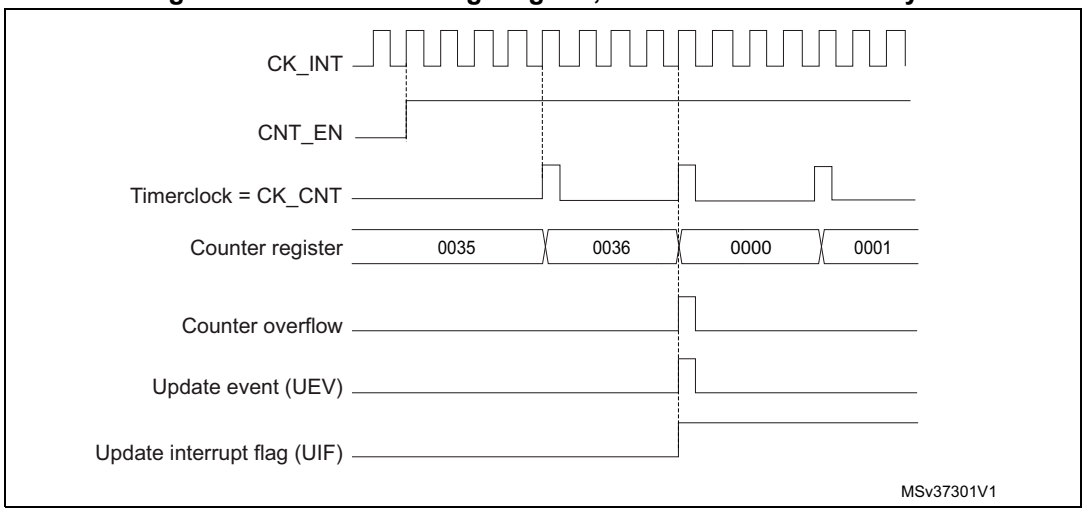


Figure 140. Counter timing diagram, internal clock divided by N

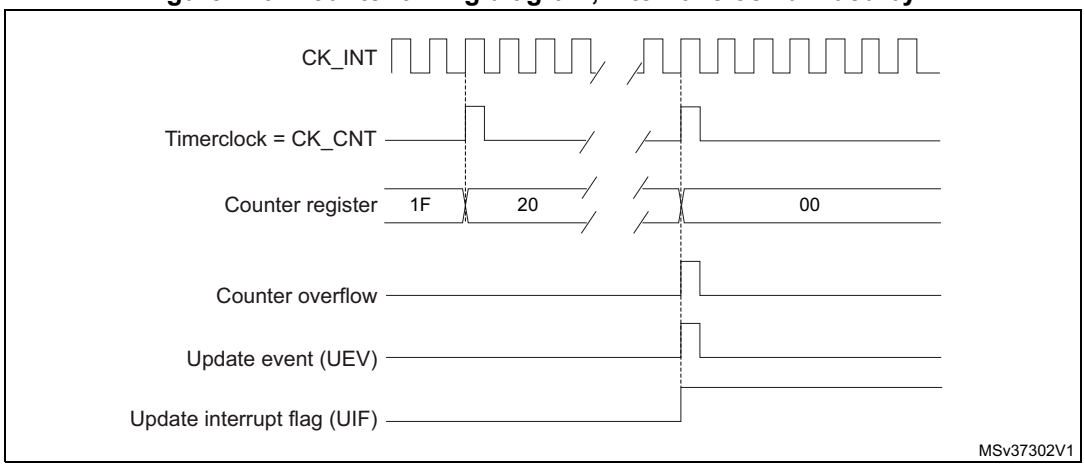


Figure 141. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)

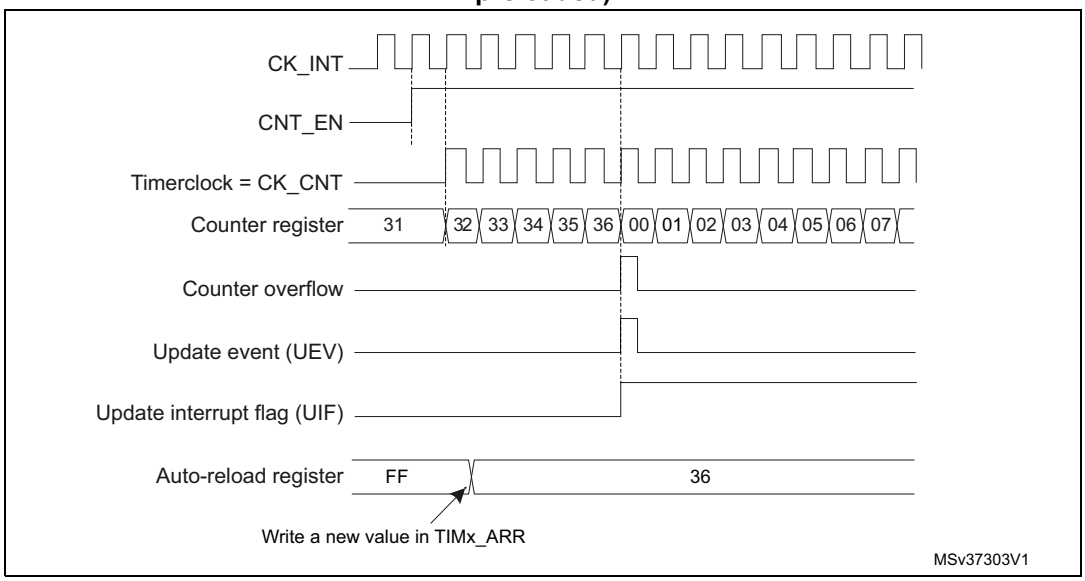
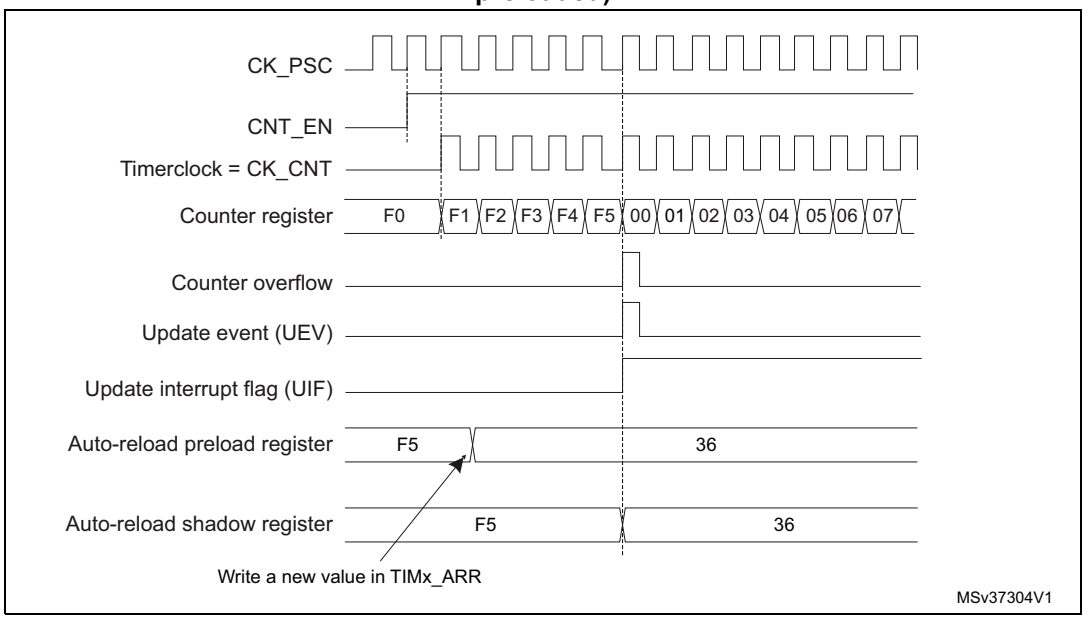


Figure 142. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)



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 generate 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 143. Counter timing diagram, internal clock divided by 1

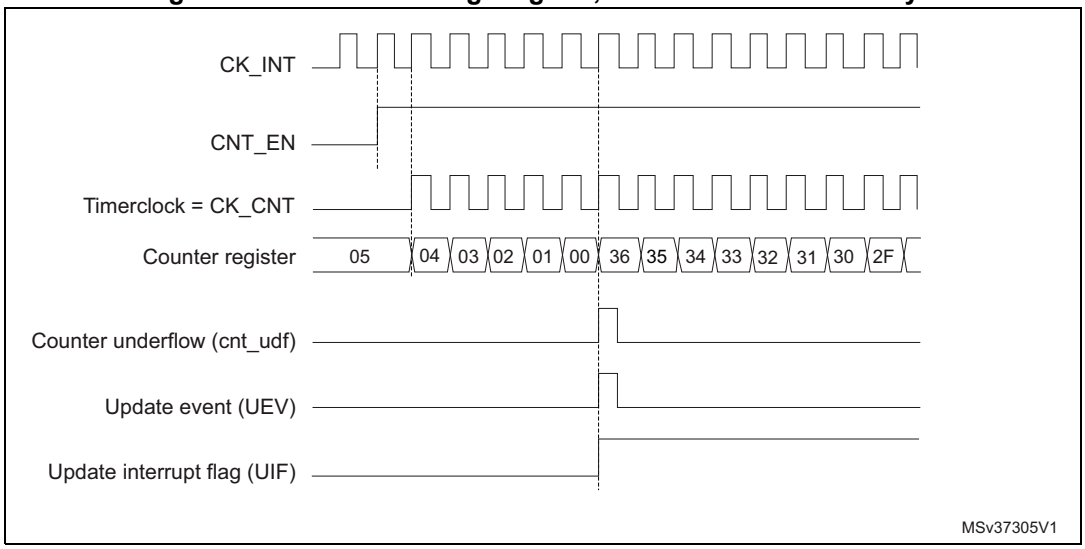


Figure 144. Counter timing diagram, internal clock divided by 2

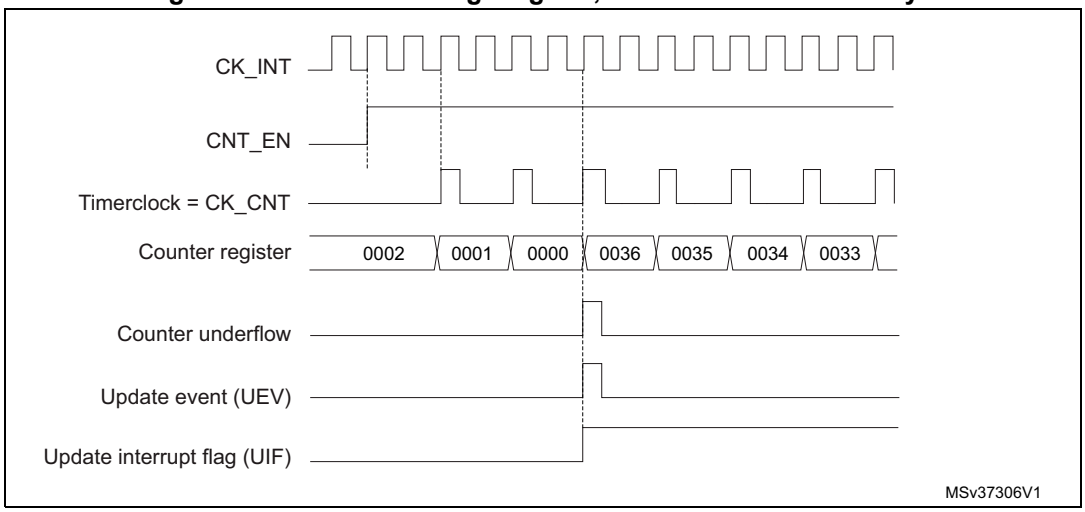


Figure 145. Counter timing diagram, internal clock divided by 4

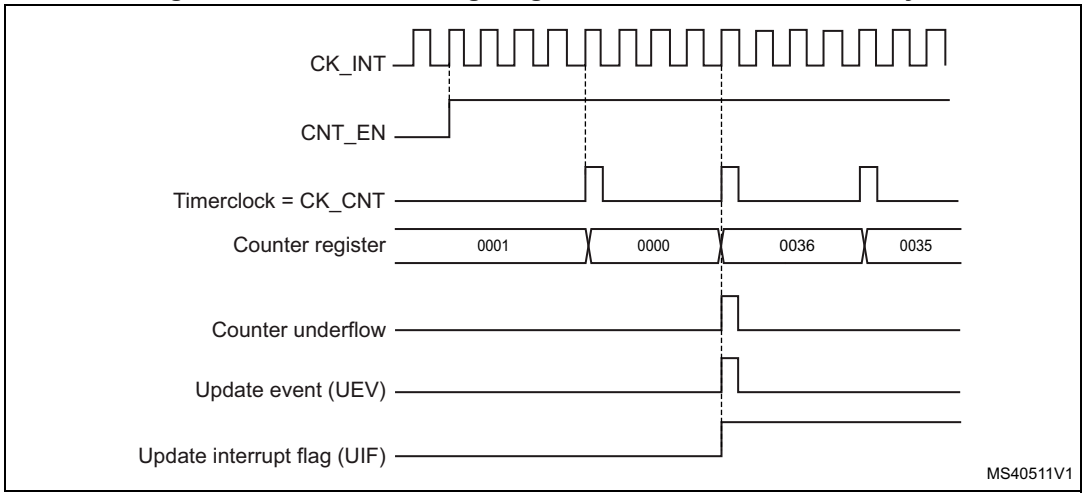


Figure 146. Counter timing diagram, internal clock divided by N

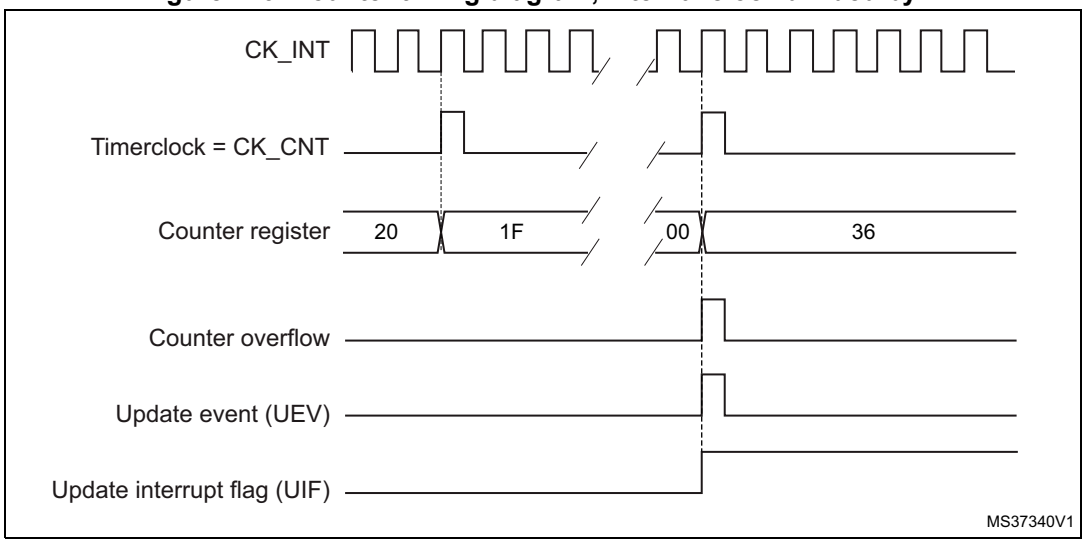
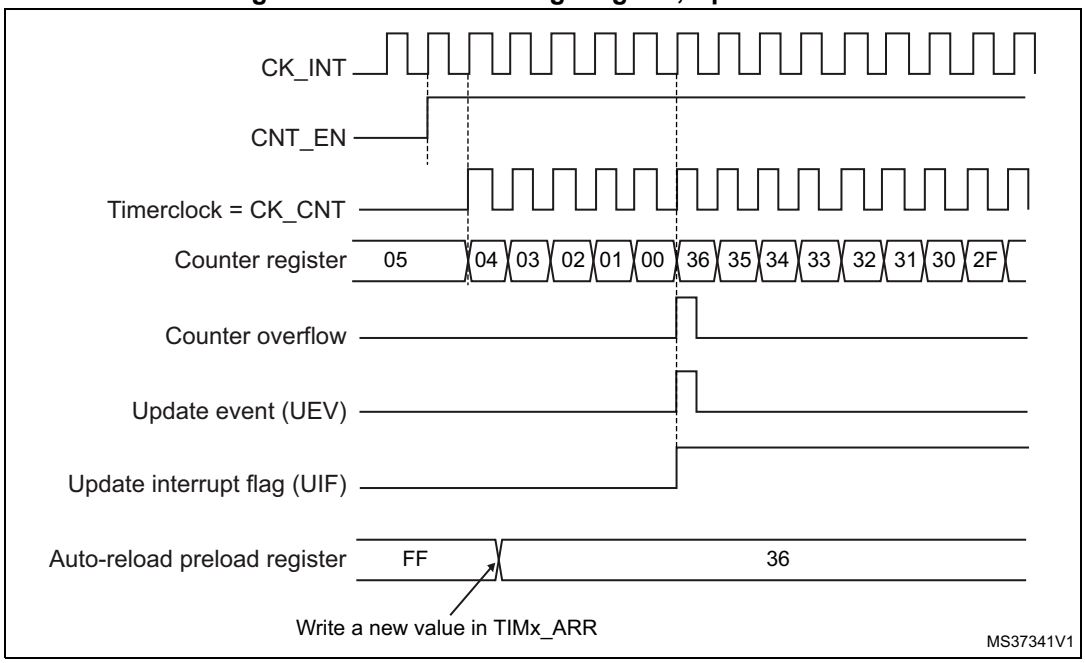


Figure 147. Counter timing diagram, Update event



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

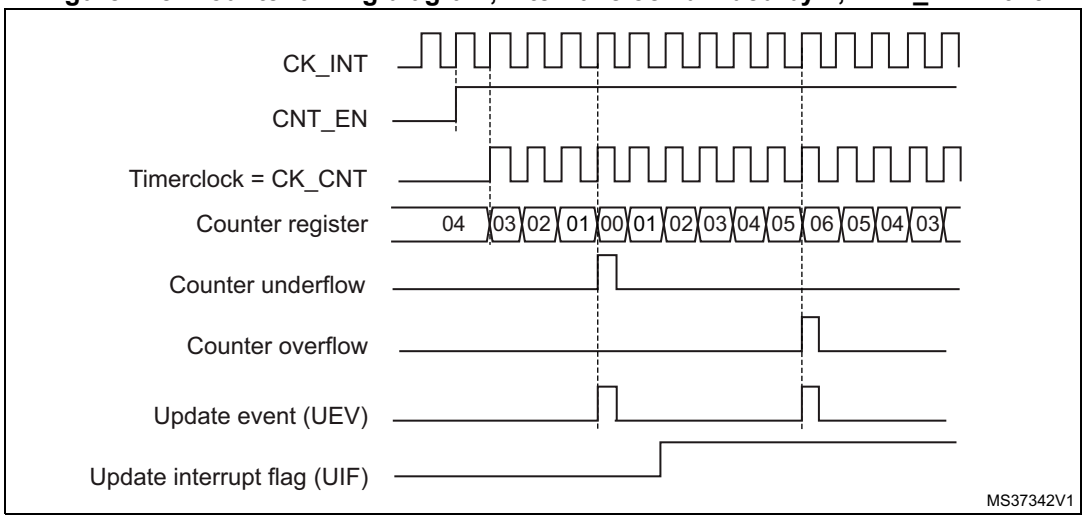
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 148. 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 18.4.1: TIMx control register 1 (TIMx_CR1)*.

Figure 149. Counter timing diagram, internal clock divided by 2

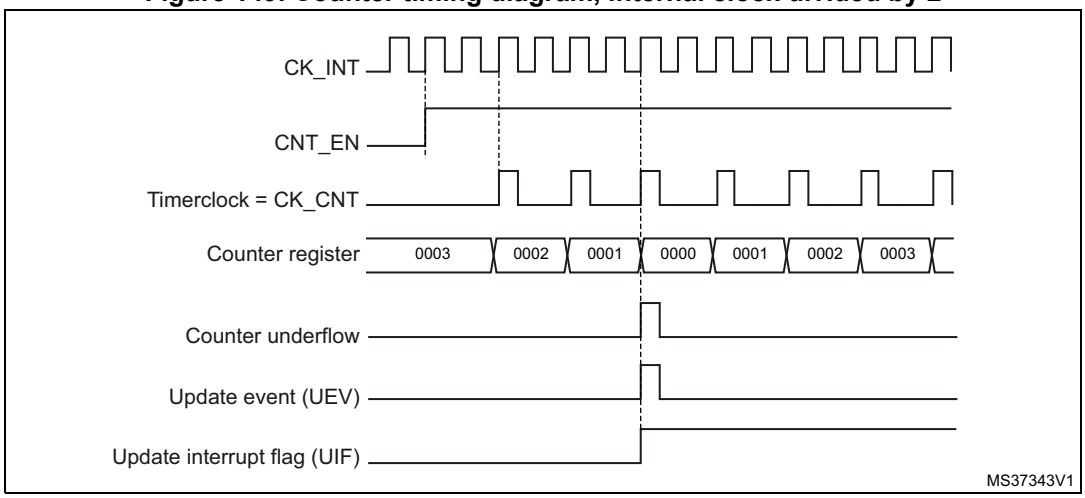
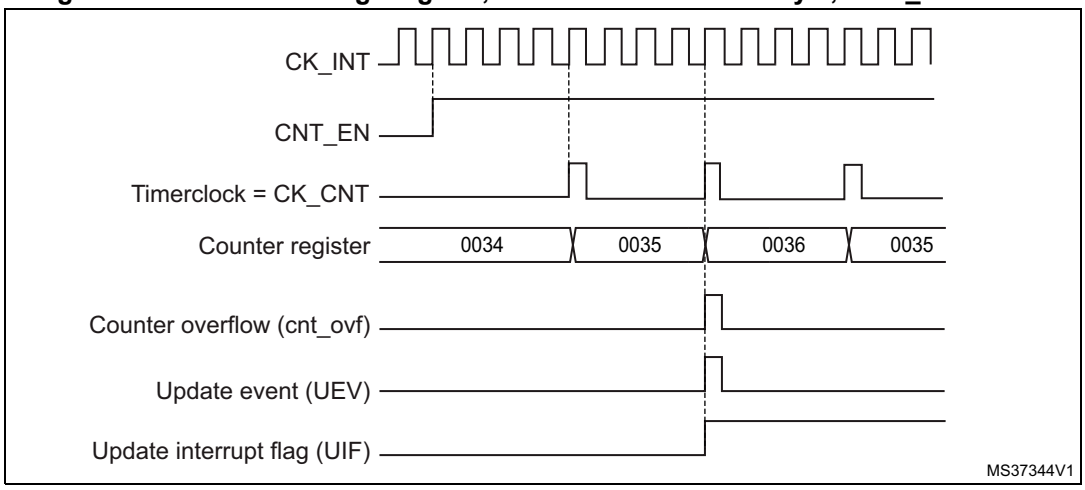


Figure 150. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36



1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 151. Counter timing diagram, internal clock divided by N

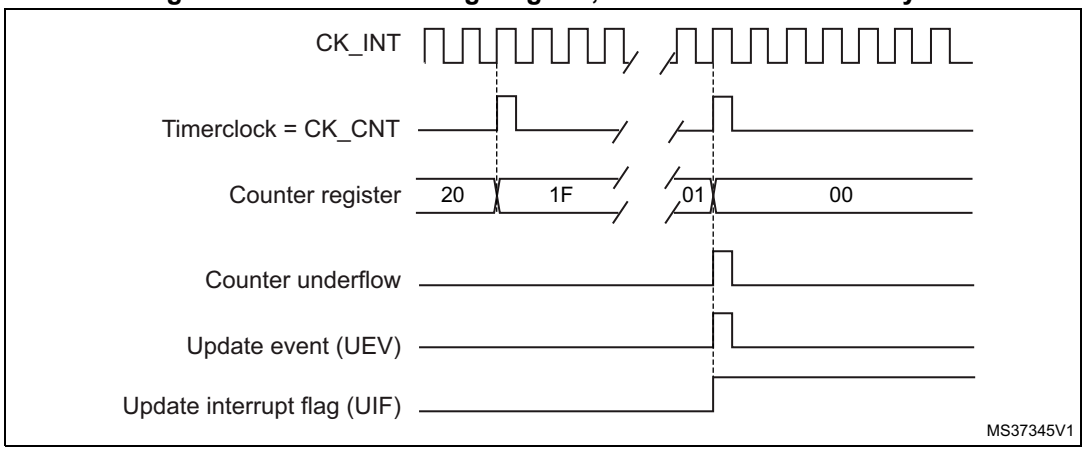


Figure 152. Counter timing diagram, Update event with ARPE=1 (counter underflow)

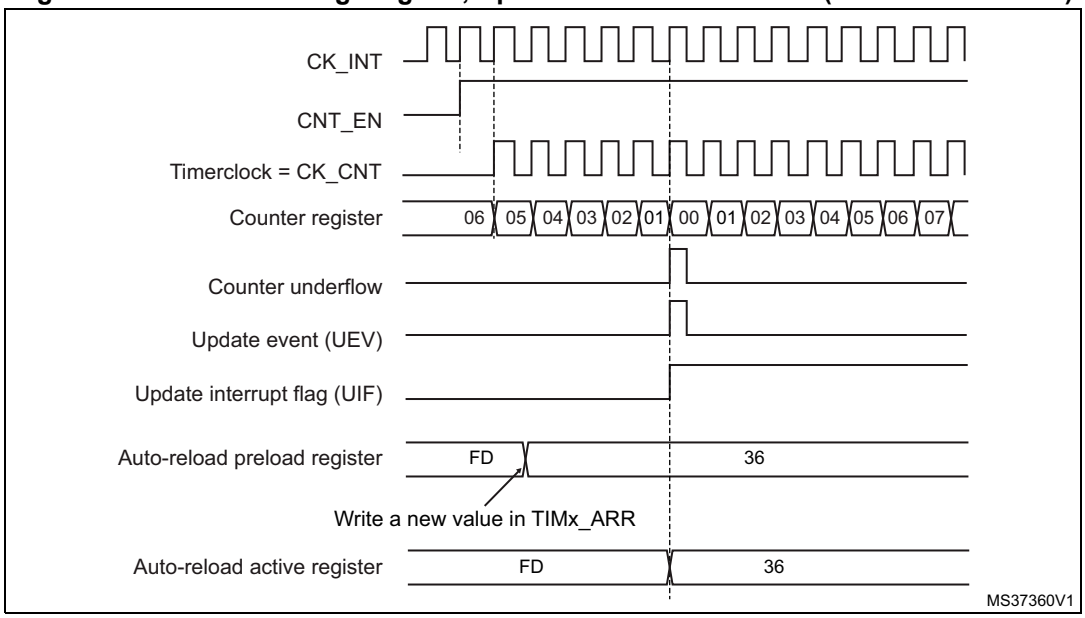
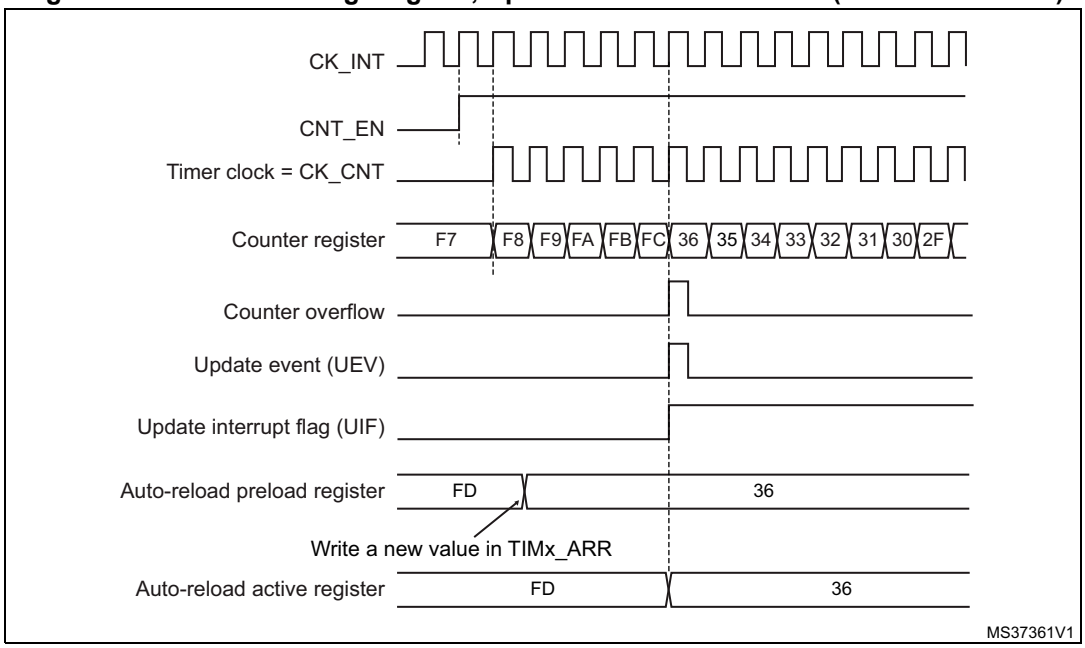


Figure 153. Counter timing diagram, Update event with ARPE=1 (counter overflow)



18.3.3 Clock selection

The counter clock can be provided by the following clock sources:

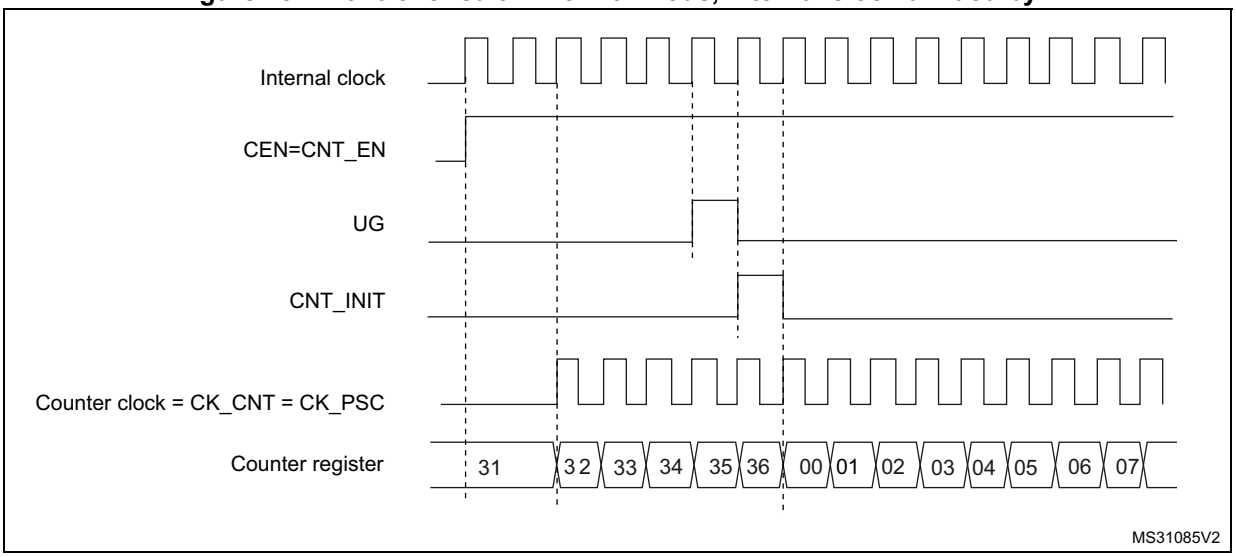
- Internal clock (CK_INT)
- External clock mode1: external input pin (TIx)
- External clock mode2: external trigger input (ETR) available on TIM2, TIM3 and TIM4 only.
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, Timer3 can be configured to act as a prescaler for Timer 2. Refer to *Using one timer as prescaler for another timer* 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 154 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

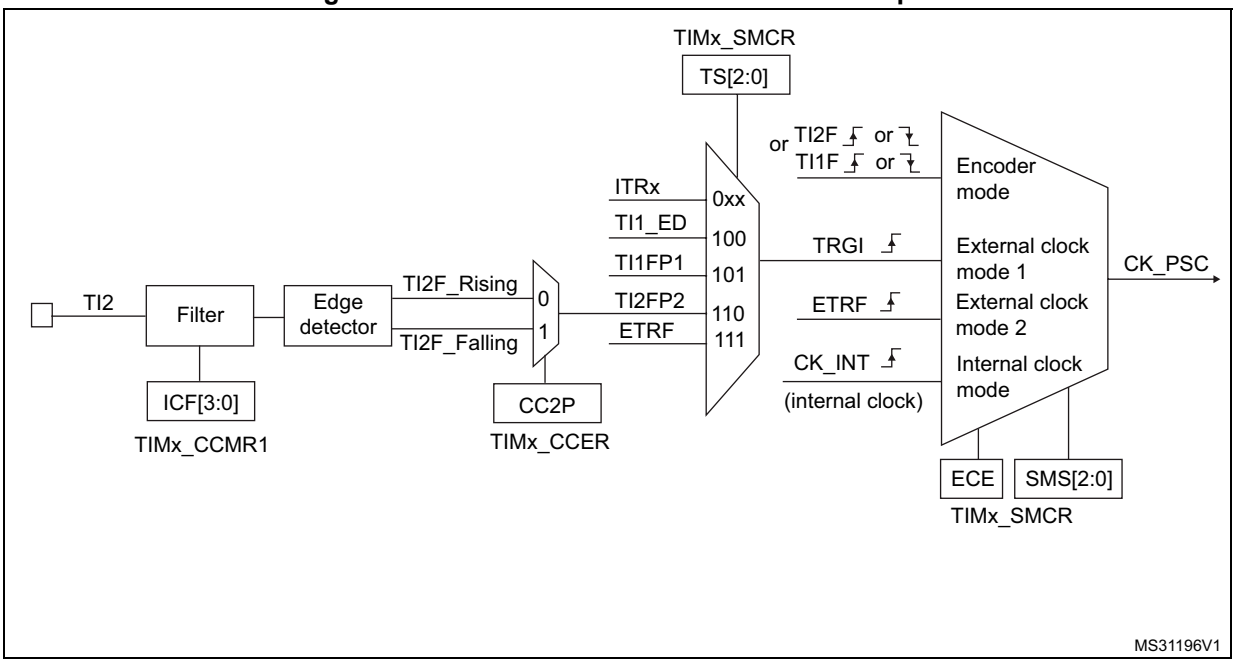
Figure 154. Control circuit in normal mode, internal clock divided by 1



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 155. 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).

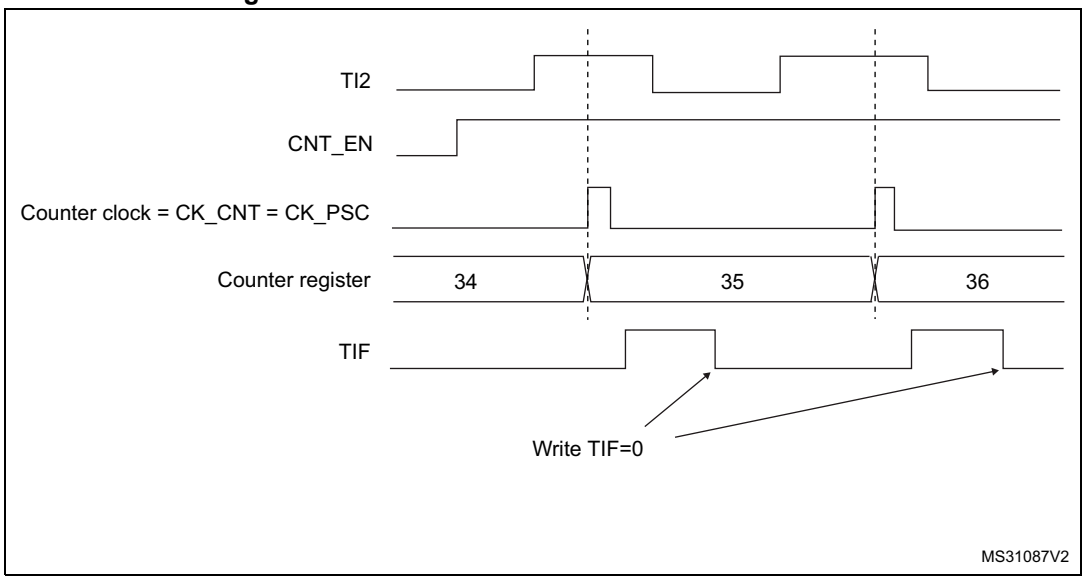
Note: The capture prescaler is not used for triggering, so there’s no need to configure it.

3. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
5. Select TI2 as the input source by writing TS=110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 156. Control circuit in external clock mode 1



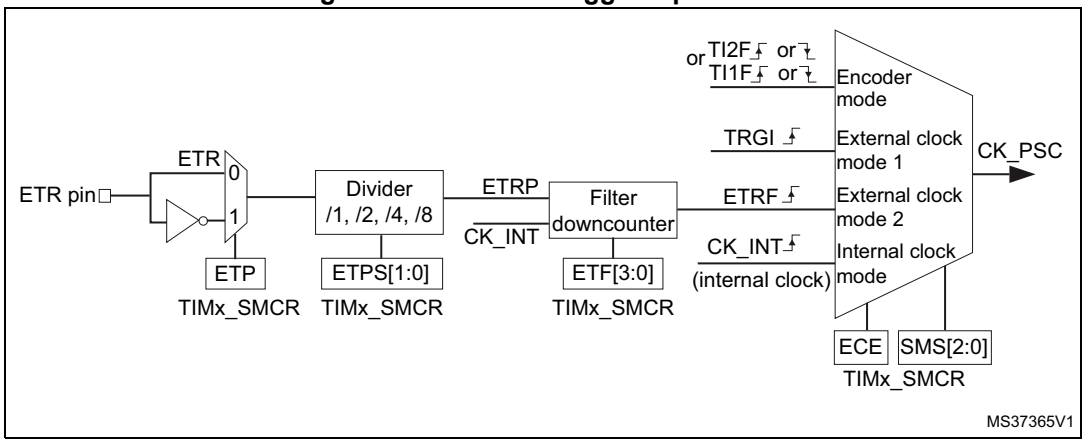
External clock source mode 2

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 157 gives an overview of the external trigger input block.

Figure 157. External trigger input block



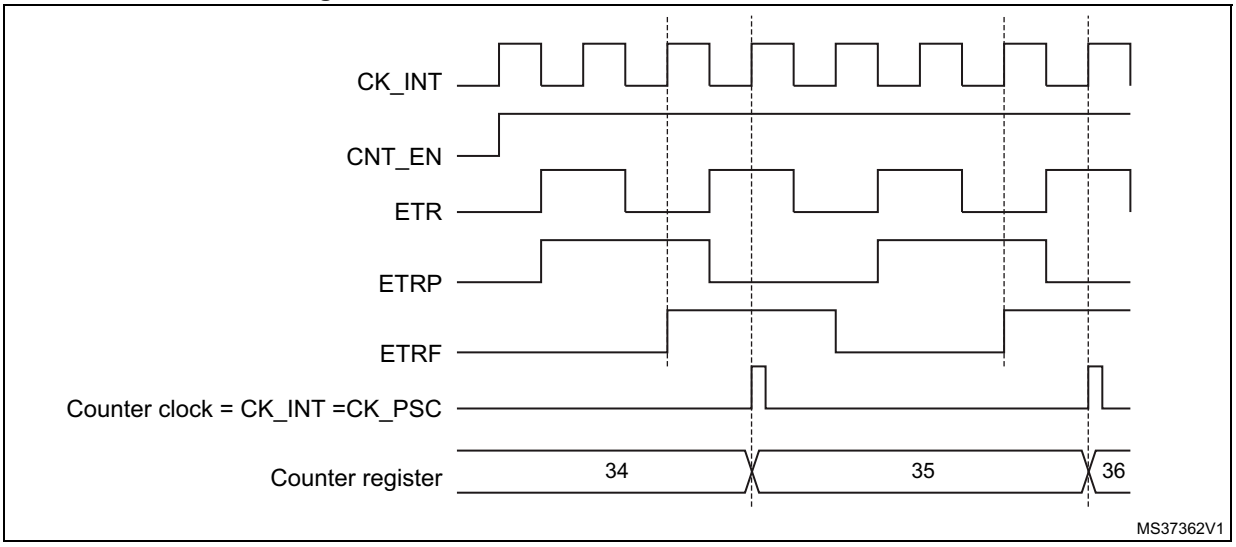
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

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 158. Control circuit in external clock mode 2

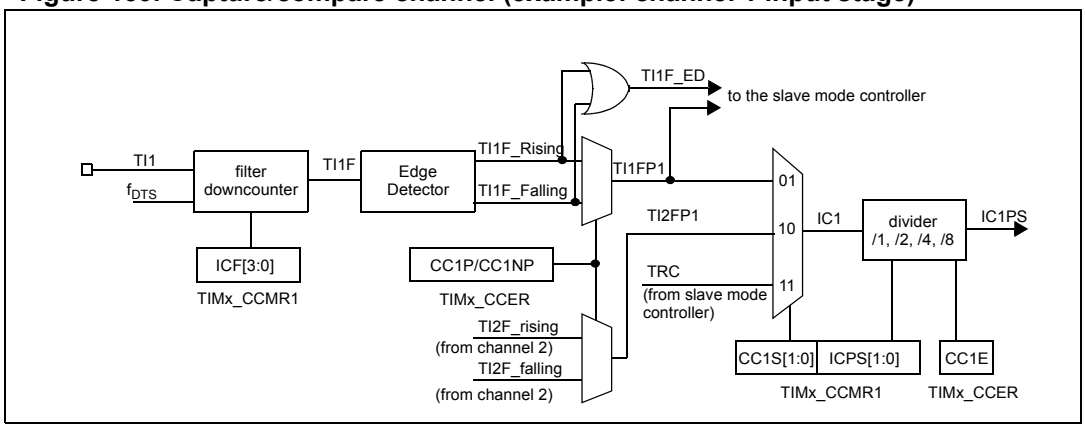


18.3.4 Capture/compare channels

Each Capture/Compare channel (see *Figure 159*) is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

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 159. 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 160. Capture/compare channel 1 main circuit

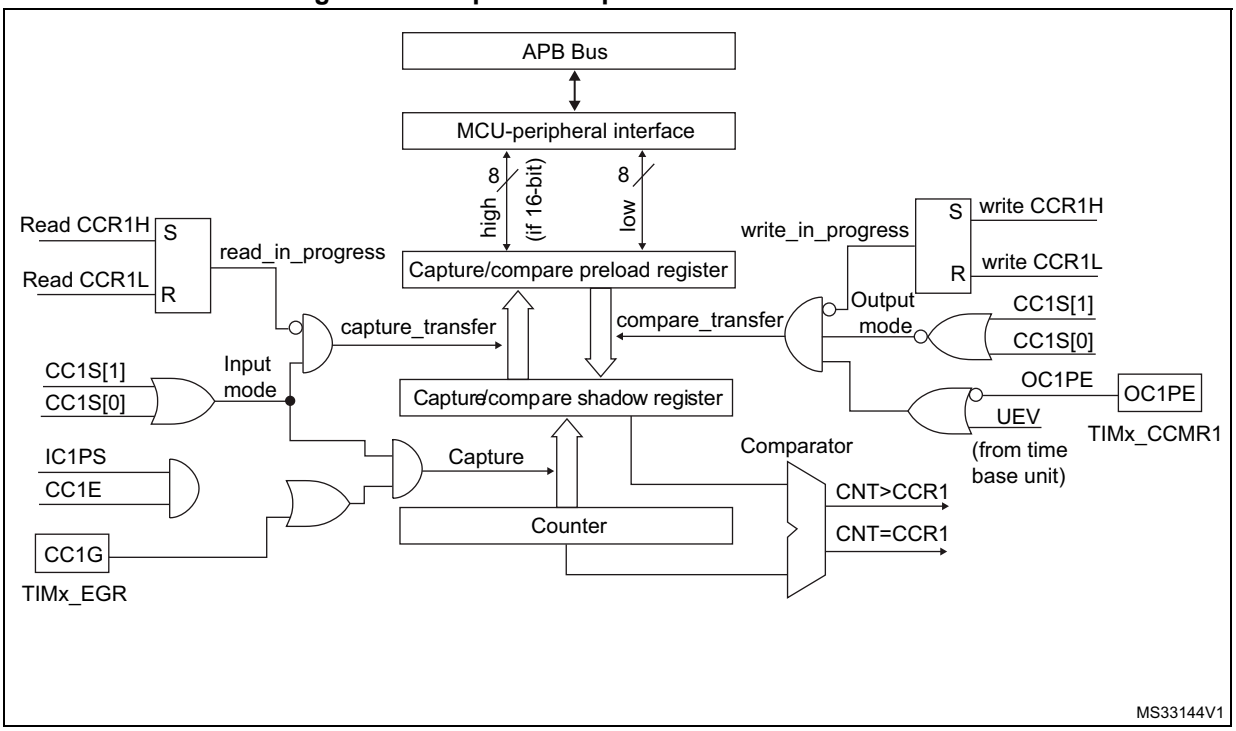
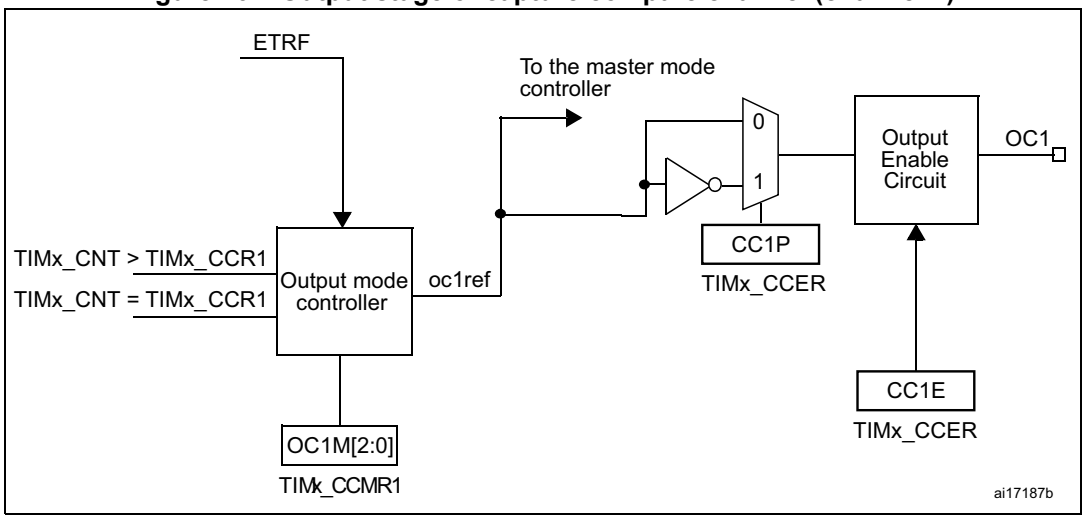


Figure 161. Output stage of capture/compare channel (channel 1)



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.

18.3.5 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 written to 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
- Program the needed input filter duration with respect to the signal connected to the timer (by programming the ICxF bits in the TIMx_CCMRx register if the input is one of the TIx inputs). Let's imagine that, when toggling, the input signal is not stable during at must five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on TI1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
- Select the edge of the active transition on the TI1 channel by writing the CC1P and CC1NP bits to 00 in the TIMx_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

18.3.6 PWM input mode

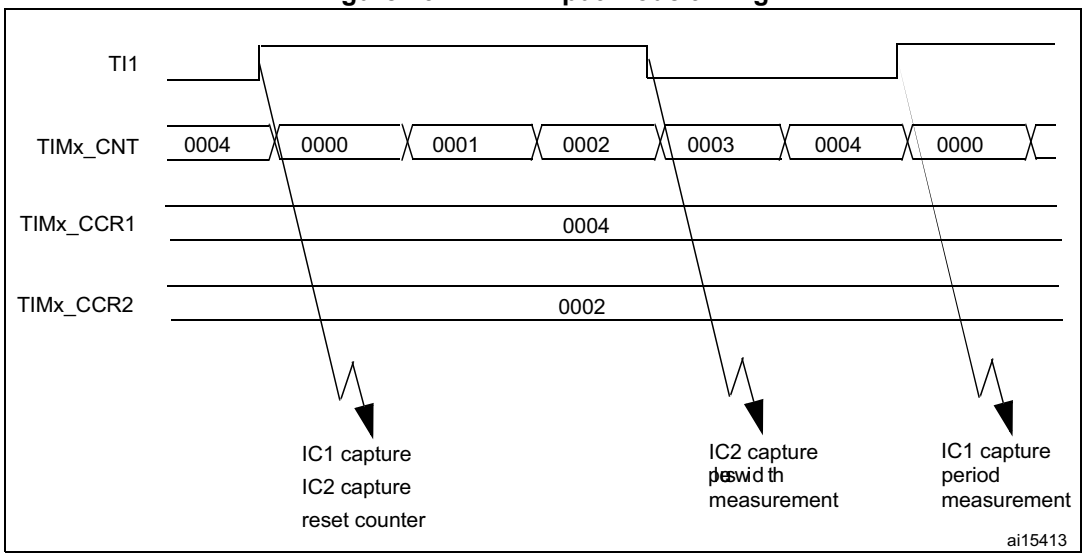
This mode is a particular case of input capture mode. The procedure is the same except:

- 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, 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):

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to '0' and the CC1NP bit to '0' (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to '1' and the CC2NP bit to '0'(active on falling edge).
- Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to '1 in the TIMx_CCER register.

Figure 162. PWM input mode timing



18.3.7 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, the user just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus ocxref is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

e.g.: CCxP=0 (OCx active high) => OCx is forced to high level.

ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare Mode section.

18.3.8 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.

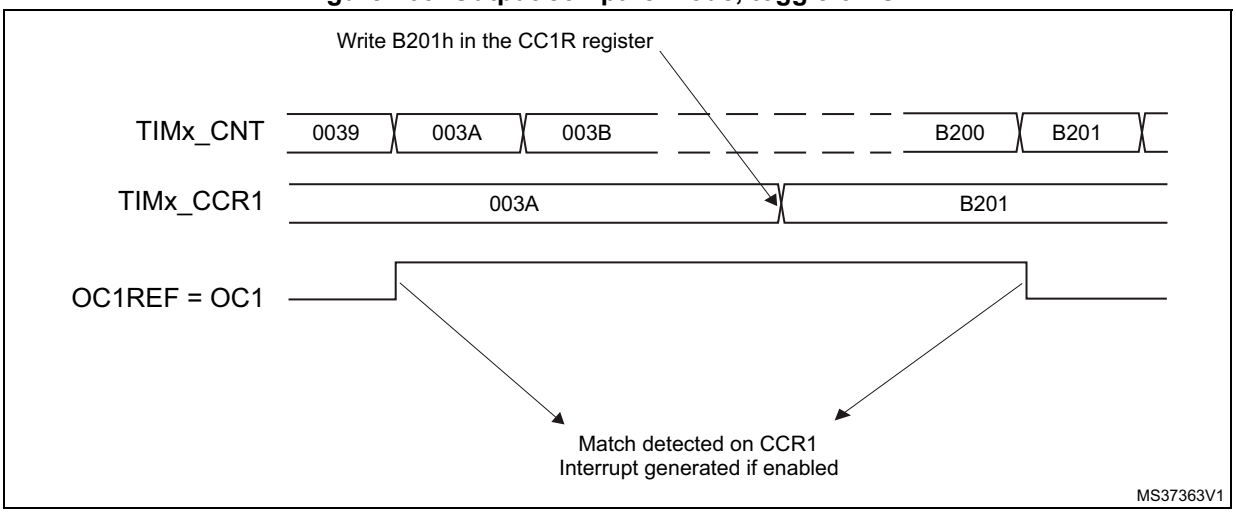
In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

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, the user 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 163*.

Figure 163. Output compare mode, toggle on OC1



18.3.9 PWM mode

Pulse width modulation mode allows generating a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing 110 (PWM mode 1) or ‘111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The user must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register 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, the user has 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≤TIMx_CNT or TIMx_CNT≤TIMx_CCRx (depending on the direction of the counter). However, to comply with the ETRF (OCREF can be cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison changes, or
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM=‘000) to one of the PWM modes (OCxM=‘110 or ‘111).

This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

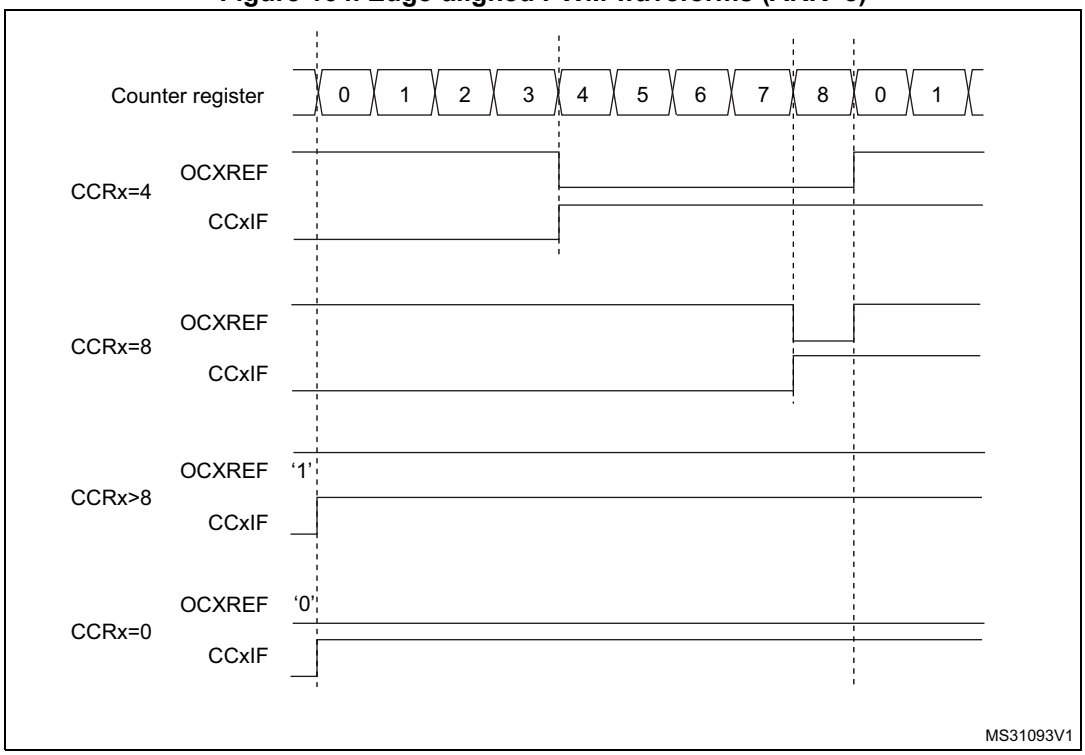
PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode*.

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 164* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 164. Edge-aligned PWM waveforms (ARR=8)



Downcounting configuration

Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to *Downcounting mode*.

In PWM mode 1, the reference signal ocxref is low as long as TIMx_CNT>TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then ocxref is held at ‘1. 0% PWM is not possible in this mode.

PWM center-aligned mode

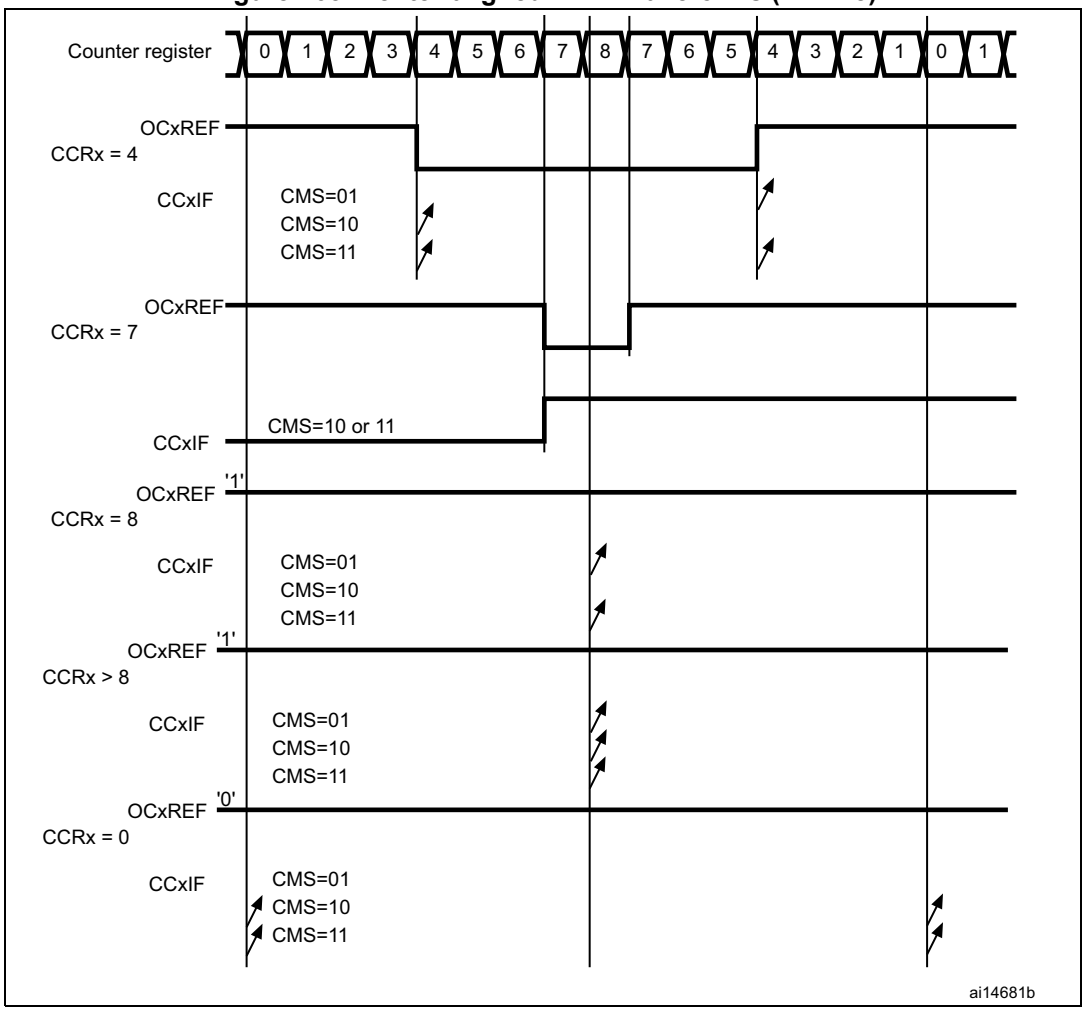
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00 (all the remaining configurations having the same effect on the ocxref/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts

up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting)*.

Figure 165 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 165. 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 the user writes 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 the user writes 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.

18.3.10 One-pulse mode

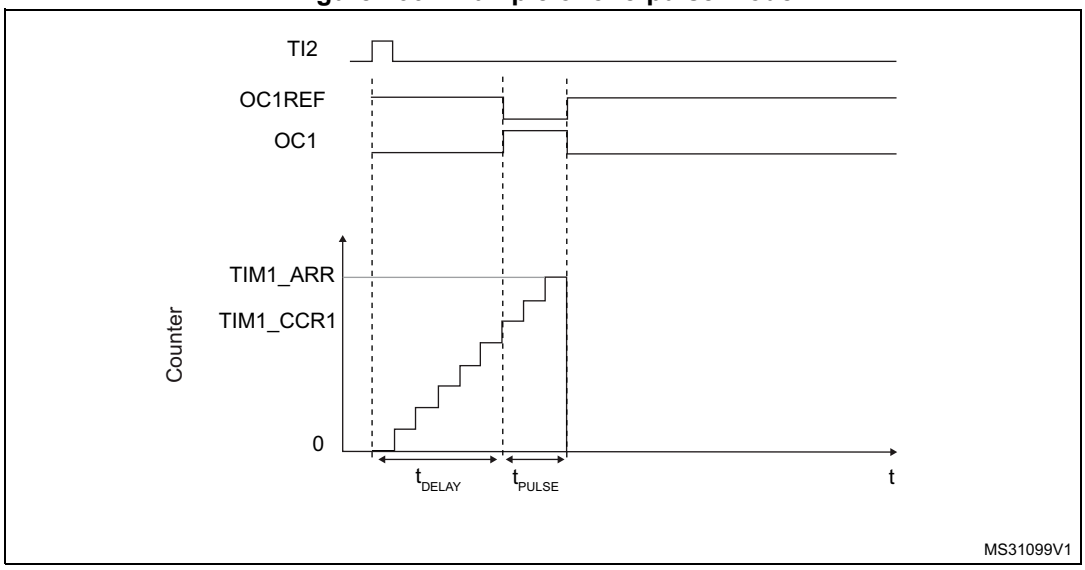
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. 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≤ARR (in particular, 0<CCRx),
- In downcounting: CNT>CCRx.

Figure 166. Example of one-pulse mode



For example the user may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

- Map TI2FP2 on TI2 by writing CC2S=01 in the TIMx_CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P=0 and CC2NP='0' in the TIMx_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=110 in the TIMx_SMCR register.
- TI2FP2 is used to start the counter by writing SMS to '110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR + 1).
- Let us say user wants 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 enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. The user 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 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.

User only wants one pulse (Single mode), so write '1 in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

To output a waveform with the minimum delay, the user 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.

18.3.11 Clearing the OCxREF signal on an external event

The OCxREF signal for a given channel can be driven Low by applying a High level to the ETRF input (OCxCE enable bit of the corresponding TIMx_CCMRx register set to '1'). The OCxREF signal remains Low until the next update event, UEV, occurs.

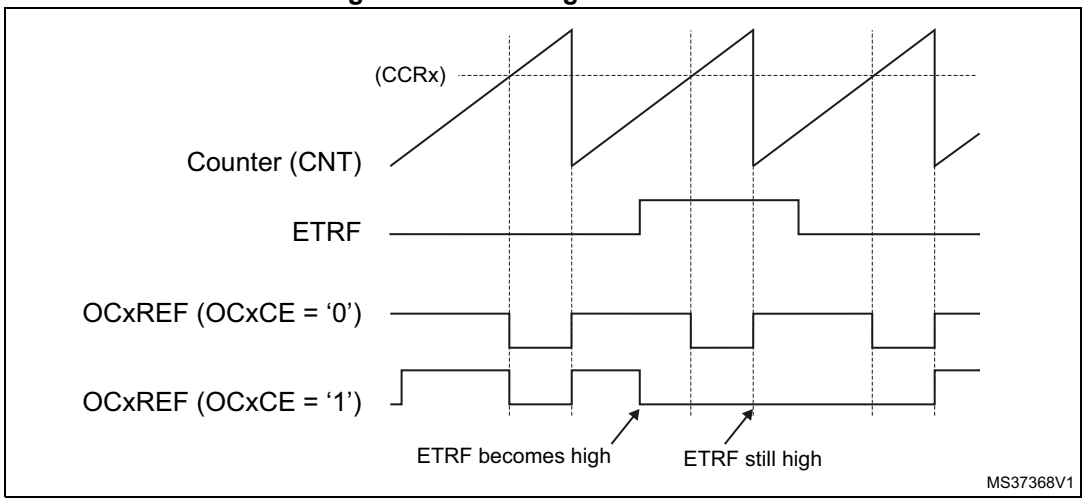
This function can only be used in output compare and PWM modes, and does not work in forced mode.

For example, the ETR 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 167 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 167. Clearing TIMx OCxREF



1. In case of a PWM with a 100% duty cycle (if CCRx>ARR), OCxREF is enabled again at the next counter overflow.

18.3.12 Encoder interface mode

To select Encoder Interface mode write SMS='001 in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS=010 if it is counting on TI1 edges only and SMS=011 if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, program the input filter as well.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to *Table 97*. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the user must configure TIMx_ARR before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the incremental encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 don't switch at the same time.

Table 97. 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 168 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S= '01' (TIMx_CCMR1 register, TI1FP1 mapped on TI1)
- CC2S= '01' (TIMx_CCMR2 register, TI2FP2 mapped on TI2)
- CC1P= '0', CC1NP = '0', IC1F ='0000' (TIMx_CCER register, TI1FP1 noninverted, TI1FP1=TI1)
- CC2P= '0', CC2NP = '0', IC2F ='0000' (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 168. Example of counter operation in encoder interface mode

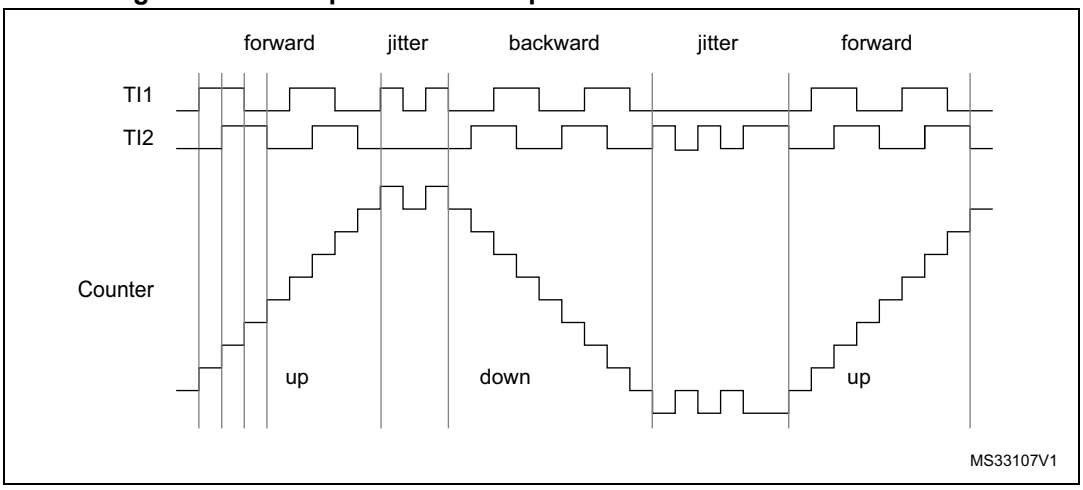
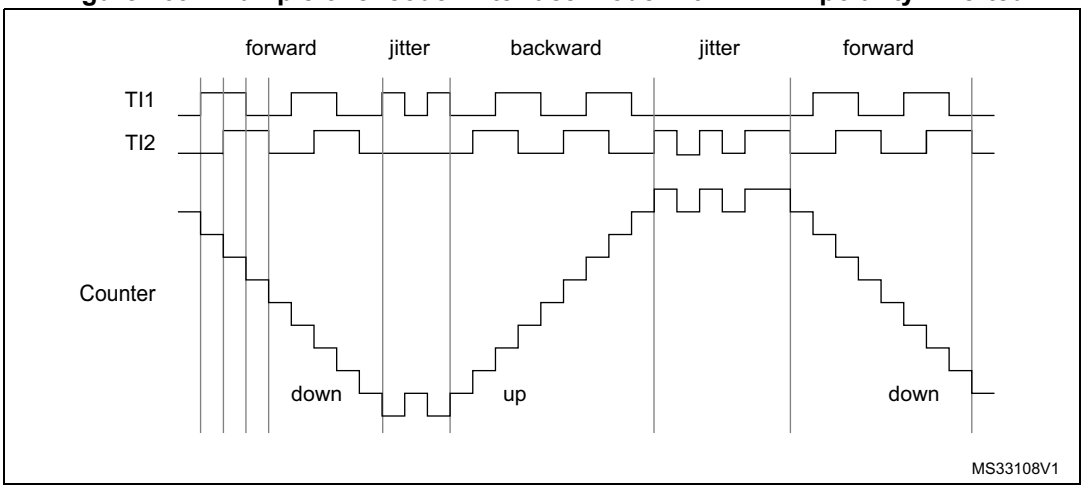


Figure 169 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

Figure 169. 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. The user 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. The user 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.

18.3.13 Timer input XOR function

The TI1S bit in the TIM_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx_CH1 to TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

18.3.14 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

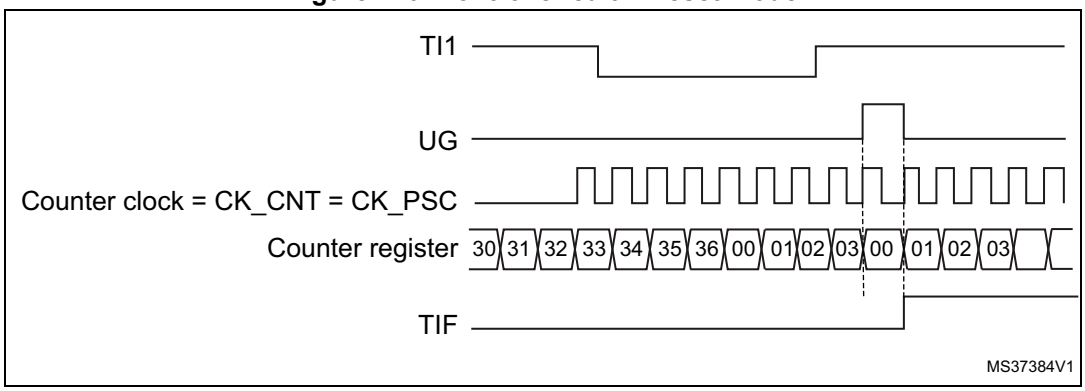
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so the user does not 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).

Figure 170 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 170. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

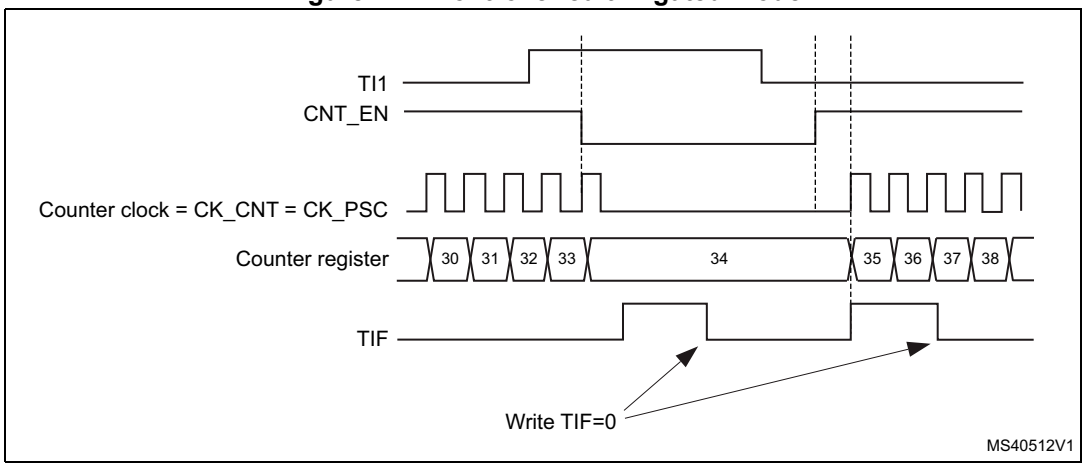
In the following example, the upcounter counts only when TI1 input is low:

- 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 the user does not need to configure it. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 171. 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.

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

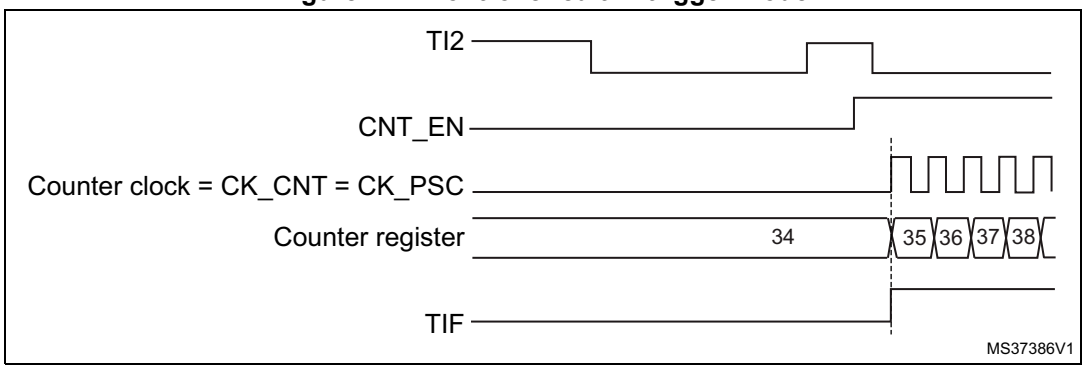
In the following example, the upcounter starts in response to a rising edge on TI2 input:

- 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 the user does not need to configure it. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P=1 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 172. Control circuit in trigger mode



Slave mode: External Clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

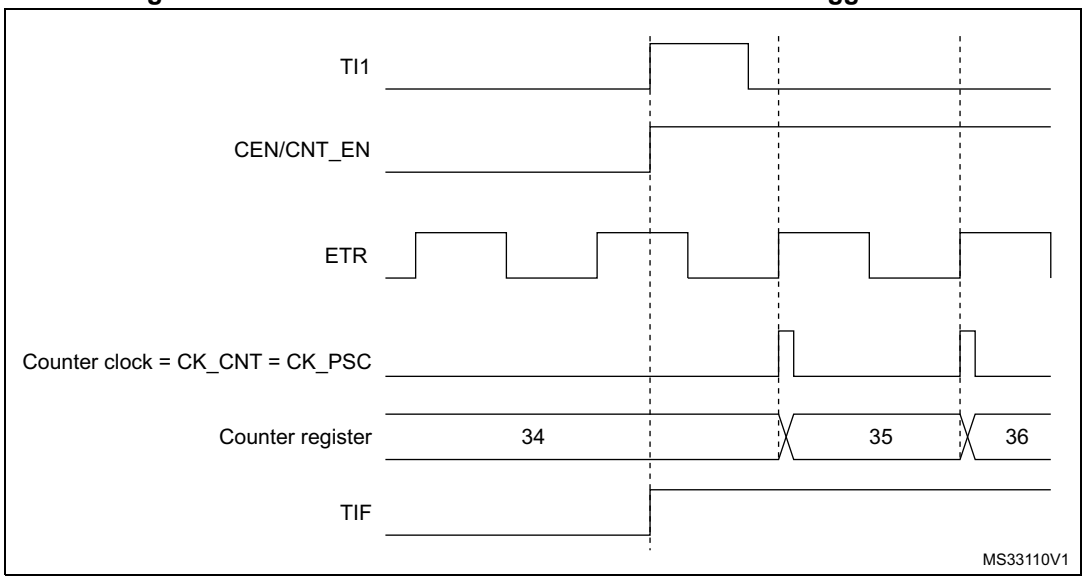
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS = 00: prescaler disabled
   - ETP = 0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01 in TIMx_CCMR1 register to select only the input capture source
   - CC1P = 0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

Figure 173. Control circuit in external clock mode 2 + trigger mode



18.3.15 Timer synchronization

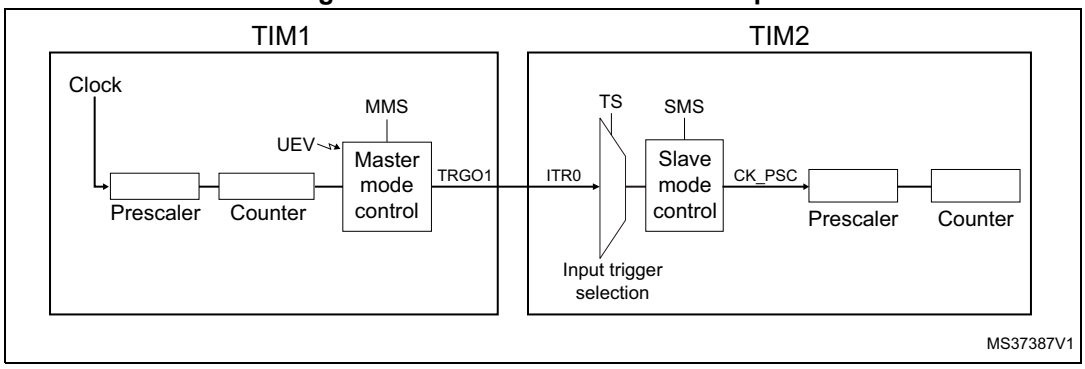
The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

Figure 174 presents an overview of the trigger selection and the master mode selection blocks.

Note: The clock of the slave timer must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Using one timer as prescaler for another timer

Figure 174. Master/Slave timer example



For example, the user can configure Timer 1 to act as a prescaler for Timer 2 (see *Figure 174*). To do this:

- Configure Timer 1 in master mode so that it outputs a periodic trigger signal on each update event UEV. If you write MMS=010 in the TIM1_CR2 register, a rising edge is output on TRGO1 each time an update event is generated.
- To connect the TRGO1 output of Timer 1 to Timer 2, Timer 2 must be configured in slave mode using ITR0 as internal trigger. You select this through the TS bits in the TIM2_SMCR register (writing TS=000).
- Then you put the slave mode controller in external clock mode 1 (write SMS=111 in the TIM2_SMCR register). This causes Timer 2 to be clocked by the rising edge of the periodic Timer 1 trigger signal (which correspond to the timer 1 counter overflow).
- Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If OCx is selected on Timer 1 as trigger output (MMS=1xx), its rising edge is used to clock the counter of timer 2.

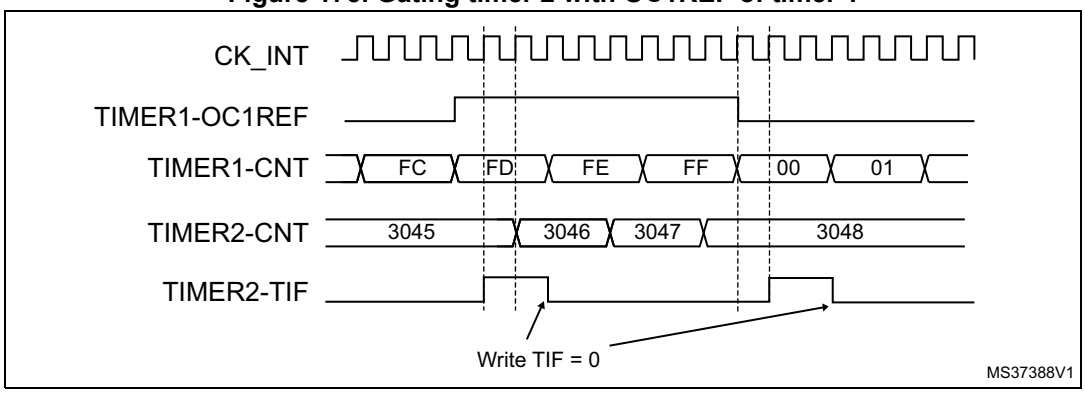
Using one timer to enable another timer

In this example, we control the enable of Timer 2 with the output compare 1 of Timer 1. Refer to *Figure 174* for connections. Timer 2 counts on the divided internal clock only when OC1REF of Timer 1 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

- Configure Timer 1 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM1_CR2 register).
- Configure the Timer 1 OC1REF waveform (TIM1_CCMR1 register).
- Configure Timer 2 to get the input trigger from Timer 1 (TS=000 in the TIM2_SMCR register).
- Configure Timer 2 in gated mode (SMS=101 in TIM2_SMCR register).
- Enable Timer 2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
- Start Timer 1 by writing ‘1 in the CEN bit (TIM1_CR1 register).

Note: The counter 2 clock is not synchronized with counter 1, this mode only affects the Timer 2 counter enable signal.

Figure 175. Gating timer 2 with OC1REF of timer 1



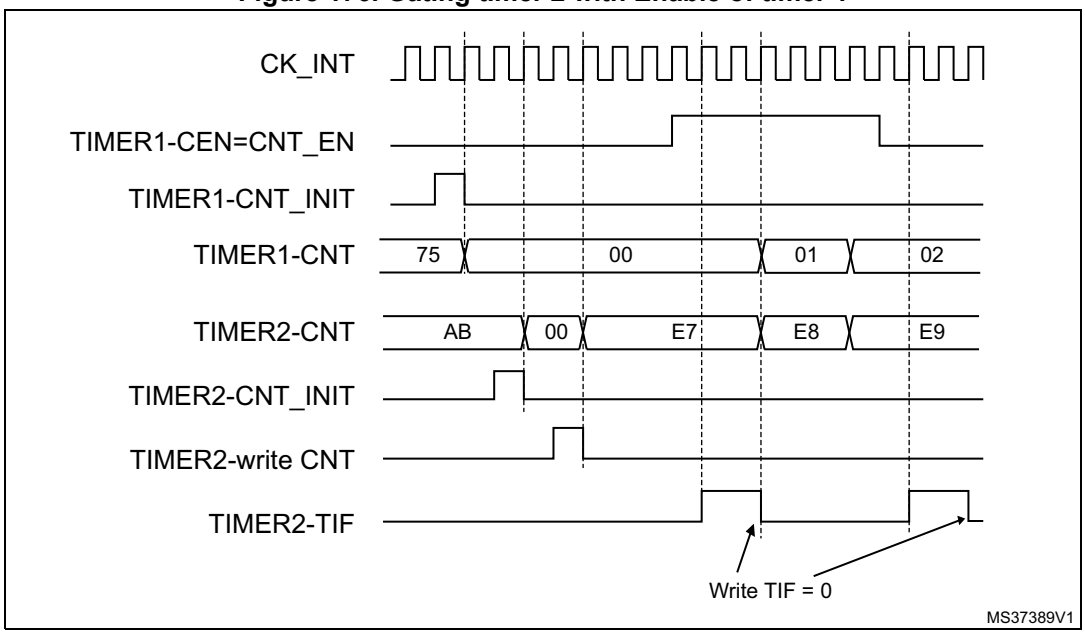
In the example in *Figure 175*, the Timer 2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting Timer 1. You can then write any value

you want in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example, we synchronize Timer 1 and Timer 2. Timer 1 is the master and starts from 0. Timer 2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. Timer 2 stops when Timer 1 is disabled by writing ‘0 to the CEN bit in the TIM1_CR1 register:

- Configure Timer 1 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM1_CR2 register).
- Configure the Timer 1 OC1REF waveform (TIM1_CCMR1 register).
- Configure Timer 2 to get the input trigger from Timer 1 (TS=000 in the TIM2_SMCR register).
- Configure Timer 2 in gated mode (SMS=101 in TIM2_SMCR register).
- Reset Timer 1 by writing ‘1 in UG bit (TIM1_EGR register).
- Reset Timer 2 by writing ‘1 in UG bit (TIM2_EGR register).
- Initialize Timer 2 to 0xE7 by writing ‘0xE7’ in the timer 2 counter (TIM2_CNTL).
- Enable Timer 2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
- Start Timer 1 by writing ‘1 in the CEN bit (TIM1_CR1 register).
- Stop Timer 1 by writing ‘0 in the CEN bit (TIM1_CR1 register).

Figure 176. Gating timer 2 with Enable of timer 1

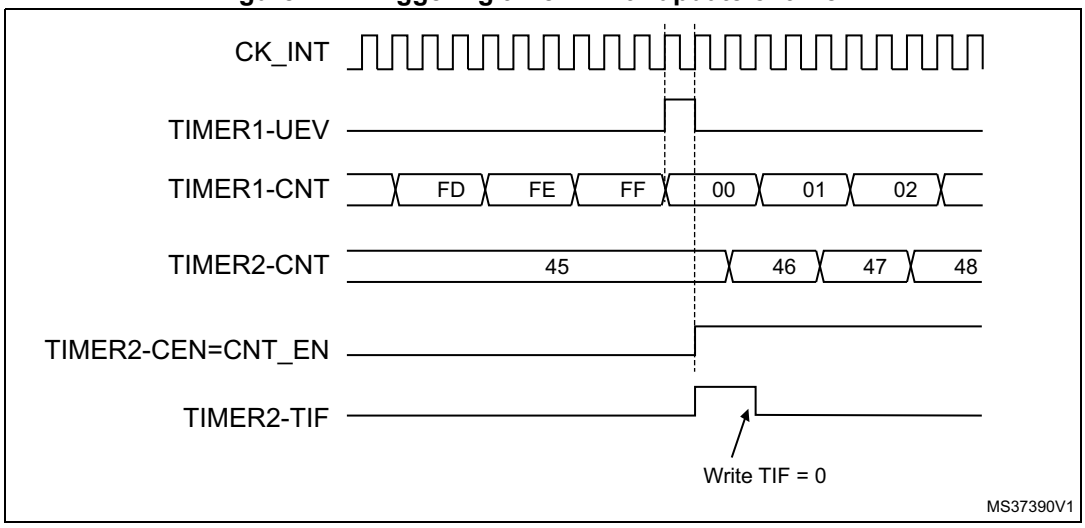


Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 1. Refer to *Figure 174* for connections. Timer 2 starts counting from its current value (which can be nonzero) 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 ($f_{CK\_CNT} = f_{CK\_INT}/3$).

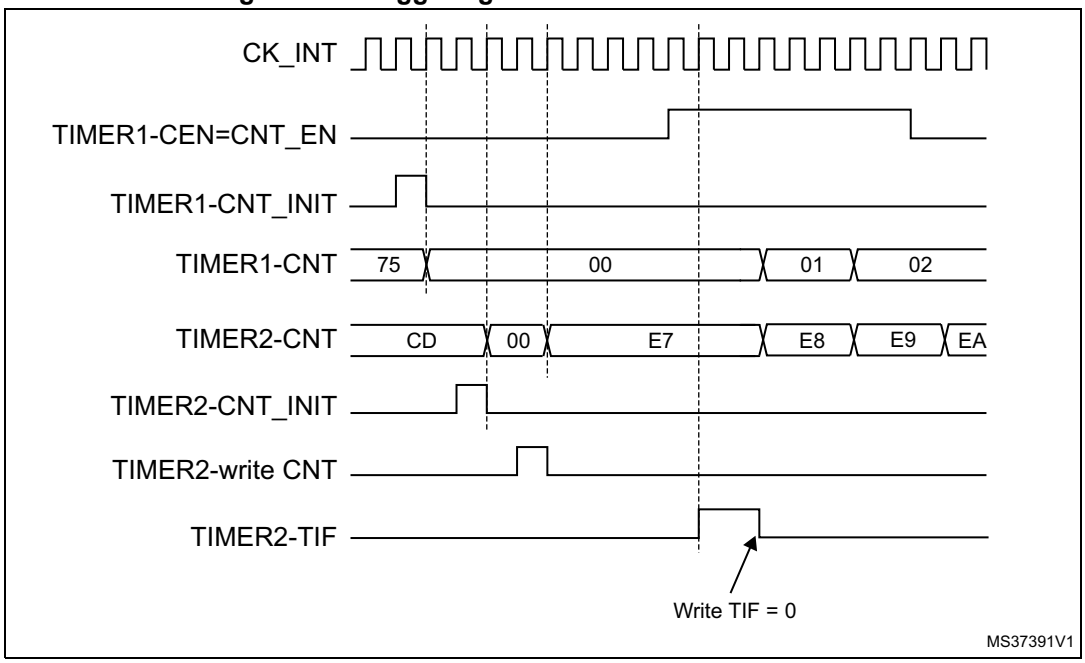
- Configure Timer 1 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM1_CR2 register).
- Configure the Timer 1 period (TIM1_ARR registers).
- Configure Timer 2 to get the input trigger from Timer 1 (TS=000 in the TIM2_SMCR register).
- Configure Timer 2 in trigger mode (SMS=110 in TIM2_SMCR register).
- Start Timer 1 by writing ‘1 in the CEN bit (TIM1_CR1 register).

Figure 177. Triggering timer 2 with update of timer 1



As in the previous example, the user can initialize both counters before starting counting. *Figure 178* shows the behavior with the same configuration as in *Figure 177* but in trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

Figure 178. Triggering timer 2 with Enable of timer 1



Starting 2 timers synchronously in response to an external trigger

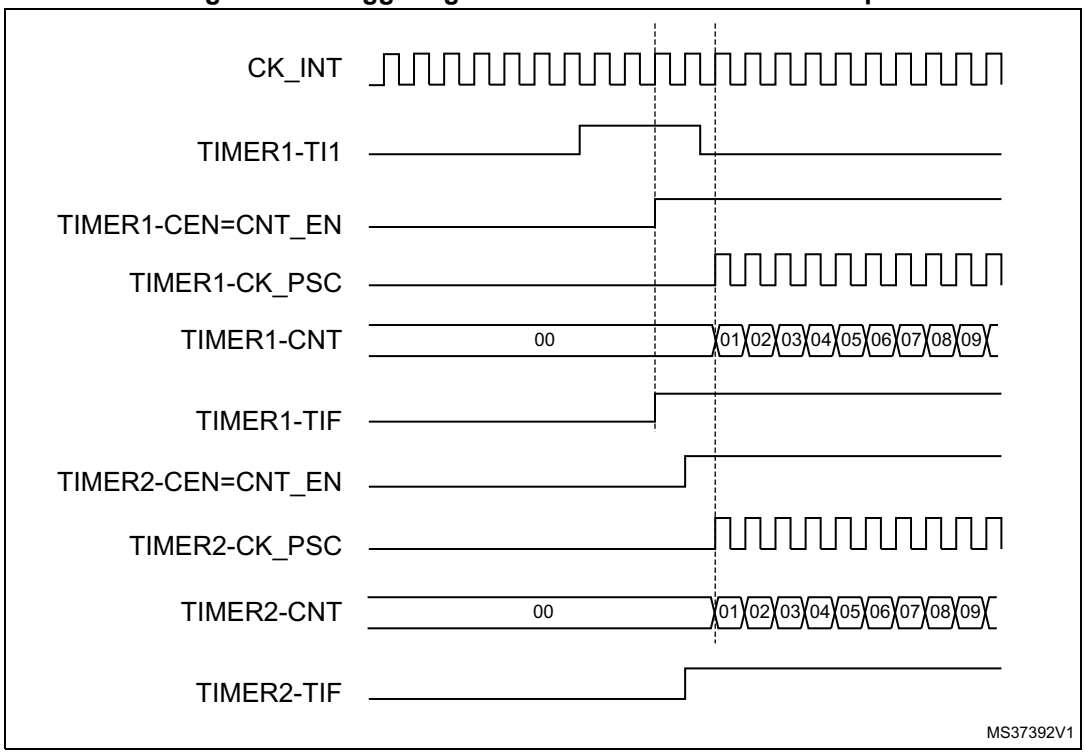
In this example, we set the enable of timer 1 when its TI1 input rises, and the enable of Timer 2 with the enable of Timer 1. Refer to *Figure 174* for connections. To ensure the counters are aligned, Timer 1 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to Timer 2):

- Configure Timer 1 master mode to send its Enable as trigger output (MMS=001 in the TIM1_CR2 register).
- Configure Timer 1 slave mode to get the input trigger from TI1 (TS=100 in the TIM1_SMCR register).
- Configure Timer 1 in trigger mode (SMS=110 in the TIM1_SMCR register).
- Configure the Timer 1 in Master/Slave mode by writing MSM=1 (TIM1_SMCR register).
- Configure Timer 2 to get the input trigger from Timer 1 (TS=000 in the TIM2_SMCR register).
- Configure Timer 2 in trigger mode (SMS=110 in the TIM2_SMCR register).

When a rising edge occurs on TI1 (Timer 1), 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 timer 1.

Figure 179. Triggering timer 1 and 2 with timer 1 TI1 input



18.3.16 Debug mode

When the microcontroller enters debug mode (Cortex®-M4 with FPU 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 38.16.2: Debug support for timers, watchdog, bxCAN and I^2C*.

18.4 TIM2 to TIM5 registers

Refer to *Section 1.1* for a list of abbreviations used in register descriptions.

The 32-bit peripheral registers have to be written by words (32 bits). All other peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

18.4.1 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
Reserved						CKD[1:0]		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **CKD**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (ETR, TIx),

00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 \times t_{CK\_INT}$
10: $t_{DTS} = 4 \times t_{CK\_INT}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

18.4.2 TIMx control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TI1S	MMS[2:0]			CCDS	Reserved		
								rw	rw	rw	rw	rw			

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

Bit 7 **TI1S**: TI1 selection

0: The TIMx_CH1 pin is connected to TI1 input

1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[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 and ADC must be enabled prior to receiving 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.

18.4.3 TIMx slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or $\overline{ETR}$ is used for trigger operations
0: ETR is noninverted, 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**: 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 98: TIMx internal trigger connection on page 632* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SMS:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

000: Slave mode disabled - if CEN = ‘1 then the prescaler is clocked directly by the internal clock.

001: Encoder mode 1 - Counter counts up/down on TI2FP1 edge depending on TI1FP2 level.

010: Encoder mode 2 - Counter counts up/down on TI1FP2 edge depending on TI2FP1 level.

011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

The clock of the slave timer must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Table 98. TIMx internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM2	TIM1_TRGO	TIM8_TRGO	TIM3_TRGO	TIM4_TRGO
TIM3	TIM1_TRGO	TIM2_TRGO	TIM5_TRGO	TIM4_TRGO
TIM4	TIM1_TRGO	TIM2_TRGO	TIM3_TRGO	TIM8_TRGO
TIM5	TIM2_TRGO	TIM3_TRGO	TIM4_TRGO	TIM8_TRGO

18.4.4 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, always read as 0

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled.
1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled.
1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled.
1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled.
1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled.
1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

18.4.5 TIMx status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

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). It is cleared by software.
0: No match
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode)

If channel CC1 is configured as input:
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred
1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

- This bit is set by hardware on an update event. It is cleared by software.
  0: No update occurred.
  1: Update interrupt pending. This bit is set by hardware when the registers are updated:
- At overflow or underflow (for TIM2 to TIM5) and if UDIS=0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.

18.4.6 TIMx event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

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

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation
refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

18.4.7 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. Take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bit 15 **OC2CE:** Output compare 2 clear enable

Bits 14:12 **OC2M[2:0]**: Output compare 2 mode

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE:** Output compare 1 clear enable

OC1CE: Output Compare 1 Clear Enable
0: OC1Ref is not affected by the ETRF input
1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 6:4 **OC1M**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.
000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).
001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.
100: Force inactive level - OC1REF is forced low.
101: Force active level - OC1REF is forced high.
110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF=‘0) as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF=1).
111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

Note: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=00 (the channel is configured in output).*

***2:** The PWM mode can be used without validating the preload register only in one-pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, IC1 is mapped on TI1.
10: CC1 channel is configured as input, IC1 is mapped on TI2.
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

Input capture mode

Bits 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).

18.4.8 TIMx capture/compare mode register 2 (TIMx_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 14:12 **OC4M**: Output compare 4 mode

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable

Bits 6:4 **OC3M**: Output compare 3 mode

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

Input capture mode

Bits 15:12 **IC4F**: Input capture 4 filter

Bits 11:10 **IC4PSC**: Input capture 4 prescaler

Bits 9:8 **CC4S**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F**: Input capture 3 filter

Bits 3:2 **IC3PSC**: Input capture 3 prescaler

Bits 1:0 **CC3S**: Capture/Compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

18.4.9 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 TI1FP1/TI2FP1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: *Capture/Compare 1 output Polarity.*
CC1 channel configured as output:
0: OC1 active high
1: OC1 active low
CC1 channel configured as input:
CC1NP/CC1P bits select TI1FP1 and TI2FP1 polarity for trigger or capture operations.
00: noninverted/rising edge
Circuit is sensitive to TIxFP1 rising edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode, encoder mode).
01: inverted/falling edge
Circuit is sensitive to TIxFP1 falling edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is inverted (trigger in gated mode, encoder mode).
10: reserved, do not use this configuration.
11: noninverted/both edges
Circuit is sensitive to both TIxFP1 rising and falling edges (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode). This configuration must not be used for encoder mode.

Bit 0 **CC1E**: *Capture/Compare 1 output enable.*
CC1 channel configured as output:
0: Off - OC1 is not active
1: On - OC1 signal is output on the corresponding output pin
CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled
1: Capture enabled

Table 99. 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 registers.

18.4.10 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[31:16] (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
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CNT[31:16]**: *High counter value (on TIM2 and TIM5).*

Bits 15:0 **CNT[15:0]**: Counter value

18.4.11 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”).

18.4.12 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] (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
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]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 18.3.1: Time-base unit* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

18.4.13 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/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

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 CC1is 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.

18.4.14 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/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

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_CCMR register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output.
If channel CC2 is configured as input:
CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

18.4.15 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)															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

Bits 31:16 **CCR3[31:16]**: High Capture/Compare 3 value (on TIM2 and TIM5).

Bits 15:0 **CCR3[15:0]**: Low Capture/Compare value

If channel CC3 is configured as output:
CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC3 output.
If channel CC3 is configured as input:
CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx_CCR3 register is read-only and cannot be programmed.

18.4.16 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)															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

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

1. if CC4 channel is configured as output (CC4S bits):
   CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.
   The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.
2. if CC4 channel is configured as input (CC4S bits in TIMx_CCMR4 register):
   CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.

18.4.17 TIMx DMA control register (TIMx_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DBL[4:0]					Reserved			DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

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.

18.4.18 TIMx DMA address for full transfer (TIMx_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) 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).

Example of how to use the DMA burst feature

In this example the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

Note: This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

18.4.19 TIM2 option register (TIM2_OR)

Address offset: 0x50

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ITR1_RMP		Reserved									
				rw	rw										

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

Bits 11:10 **ITR1_RMP:** Internal trigger 1 remap
Set and cleared by software.
00: TIM8_TRGOUT
01: PTP trigger output is connected to TIM2_ITR1
10: OTG FS SOF is connected to the TIM2_ITR1 input
11: OTG HS SOF is connected to the TIM2_ITR1 input

Bits 9:0 Reserved, must be kept at reset value.

18.4.20 TIM5 option register (TIM5_OR)

Address offset: 0x50

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TI4_RMP		Reserved					
								rw	rw						

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

Bits 7:6 **TI4_RMP:** Timer Input 4 remap
Set and cleared by software.
00: TIM5 Channel4 is connected to the GPIO: Refer to the Alternate function mapping table in the datasheets.
01: the LSI internal clock is connected to the TIM5_CH4 input for calibration purposes
10: the LSE internal clock is connected to the TIM5_CH4 input for calibration purposes
11: the RTC wakeup interrupt is connected to TIM5_CH4 input for calibration purposes. Wakeup interrupt should be enabled.

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

18.4.21 TIMx register map

TIMx registers are mapped as described in the table below:

Table 100. TIM2 to TIM5 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Reserved																						CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x04	**TIMx_CR2**	Reserved																								TI1S	MMS[2:0]			CCDS	Reserved		
	Reset value																									0	0	0	0	0			
0x08	**TIMx_SMCR**	Reserved																ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	**TIMx_DIER**	Reserved																	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Reserved	TIE	Reserved	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0		0		0	0	0	0	0
0x10	**TIMx_SR**	Reserved																			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Reserved	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0			0		0	0	0	0	0
0x14	**TIMx_EGR**	Reserved																									TG	Reserved	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x18	**TIMx_CCMR1** Output Compare mode	Reserved																OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	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	Reserved																IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIMx_CCMR2** Output Compare mode	Reserved																O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Input Capture mode	Reserved																IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIMx_CCER**	Reserved																CC4NP	Reserved	CC4P	CC4E	CC3NP	Reserved	CC3P	CC3E	CC2NP	Reserved	CC2P	CC2E	CC1NP	Reserved	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0
0x24	**TIMx_CNT**	CNT[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 100. TIM2 to TIM5 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
0x2C	**TIMx_ARR**	ARR[31:16] (TIM2 and TIM5 only, reserved on the other timers)																ARR[15:0]															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	Reserved																																
0x34	**TIMx_CCR1**	CCR1[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIMx_CCR2**	CCR2[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR2[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**TIMx_CCR3**	CCR3[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR3[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**TIMx_CCR4**	CCR4[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR4[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	Reserved																																
0x48	**TIMx_DCR**	Reserved																			DBL[4:0]					Reserved		DBA[4:0]					
	Reset value																				0	0	0	0	0			0	0	0	0	0	0
0x4C	**TIMx_DMAR**	Reserved																DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**TIM2_OR**	Reserved																Reserved				ITR1_RMP		Reserved									
	Reset value																					0	0										
0x50	**TIM5_OR**	Reserved																Reserved											IT4_RMP	Reserved				
	Reset value																												0	0			

Refer to *Section 2.3: Memory map* for the register boundary addresses.

19 General-purpose timers (TIM9 to TIM14)

This section applies to the whole STM32F4xx family, unless otherwise specified.

19.1 TIM9 to TIM14 introduction

The TIM9 to TIM14 general-purpose timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM9 to TIM14 timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 19.3.12*.

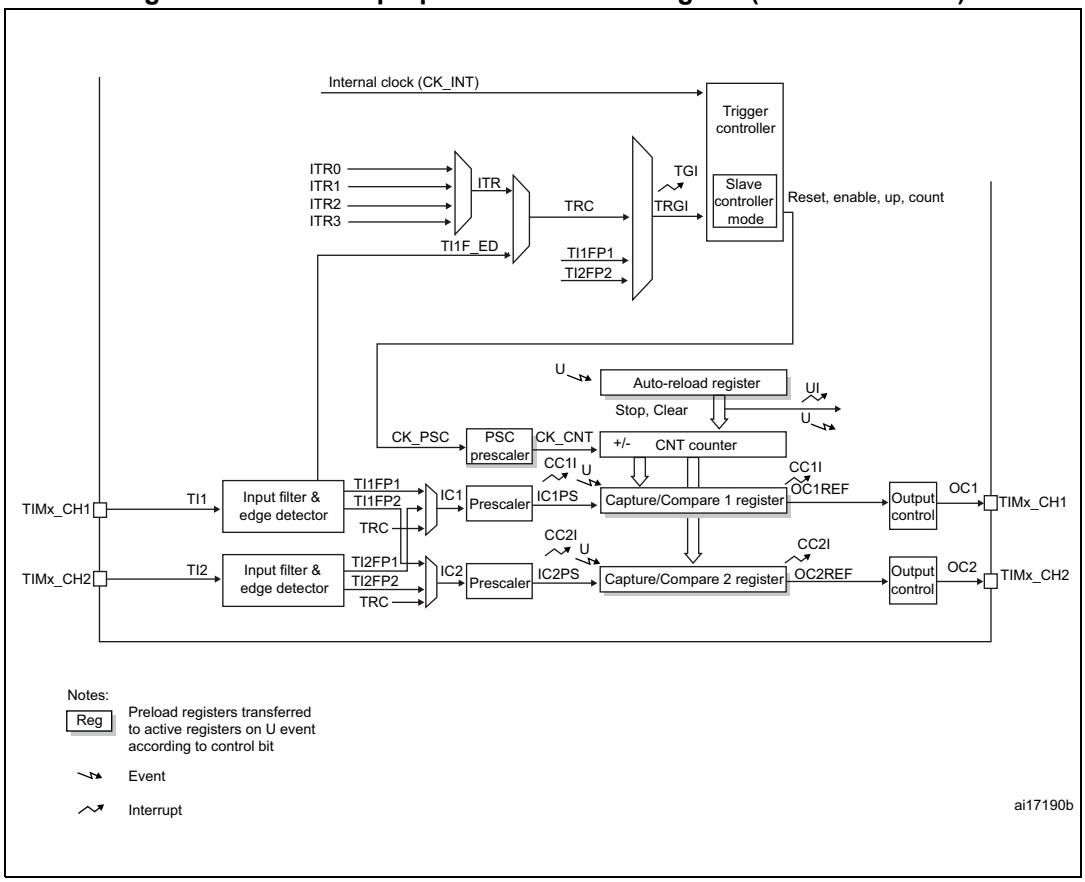
19.2 TIM9 to TIM14 main features

19.2.1 TIM9/TIM12 main features

The features of the TIM9 to TIM14 general-purpose timers include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal trigger)
  - Trigger event (counter start, stop, initialization or count by internal trigger)
  - Input capture
  - Output compare

Figure 180. General-purpose timer block diagram (TIM9 and TIM12)

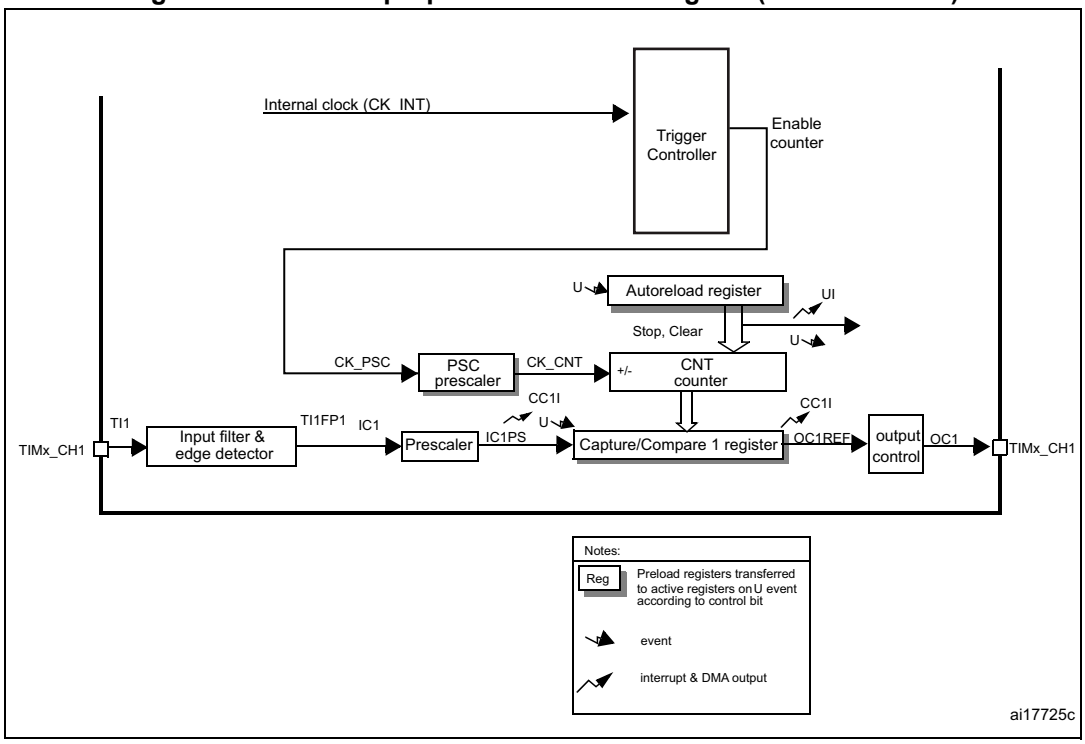


19.2.2 TIM10/TIM11 and TIM13/TIM14 main features

The features of general-purpose timers TIM10/TIM11 and TIM13/TIM14 include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- independent channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare

Figure 181. General-purpose timer block diagram (TIM10/11/13/14)



19.3 TIM9 to TIM14 functional description

19.3.1 Time-base unit

The main block of the timer is a 16-bit counter with its related auto-reload register. The counters counts up.

The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in details for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (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 182 and *Figure 183* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 182. Counter timing diagram with prescaler division change from 1 to 2

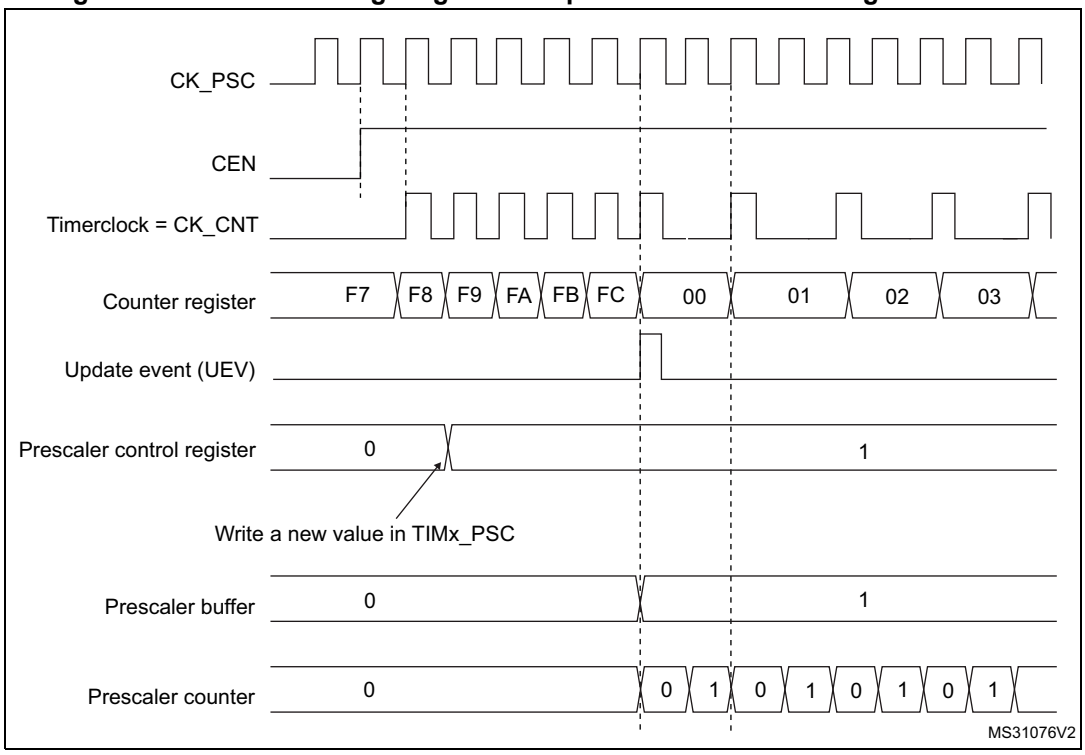
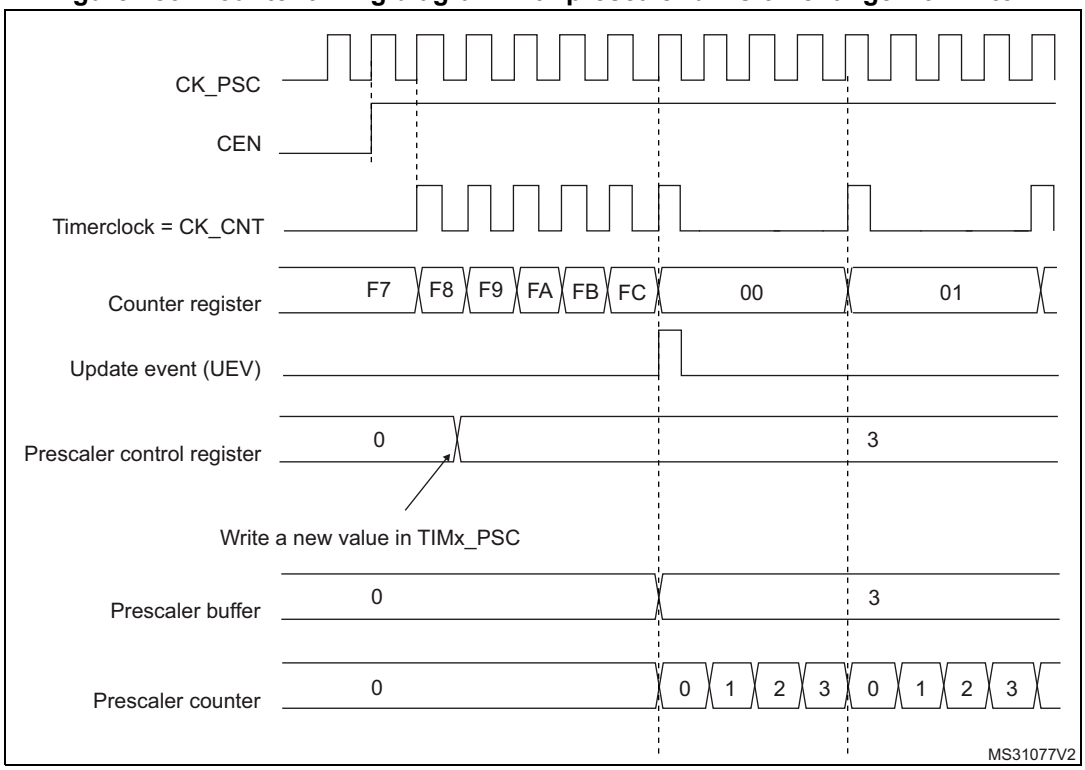


Figure 183. Counter timing diagram with prescaler division change from 1 to 4



19.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller on TIM9 and TIM12) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 184. Counter timing diagram, internal clock divided by 1

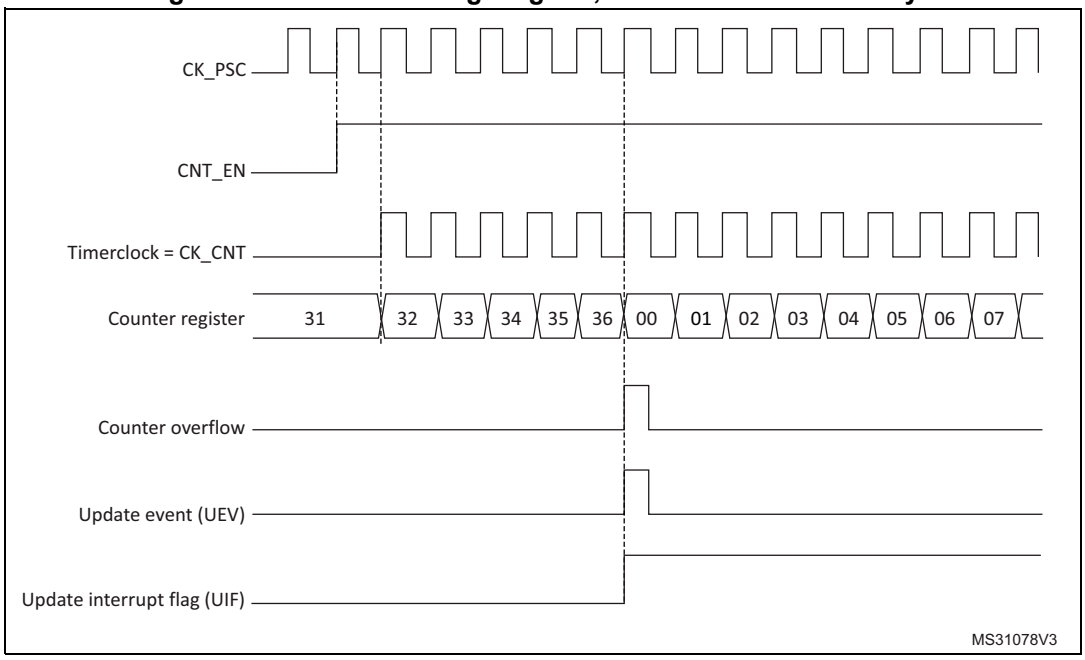


Figure 185. Counter timing diagram, internal clock divided by 2

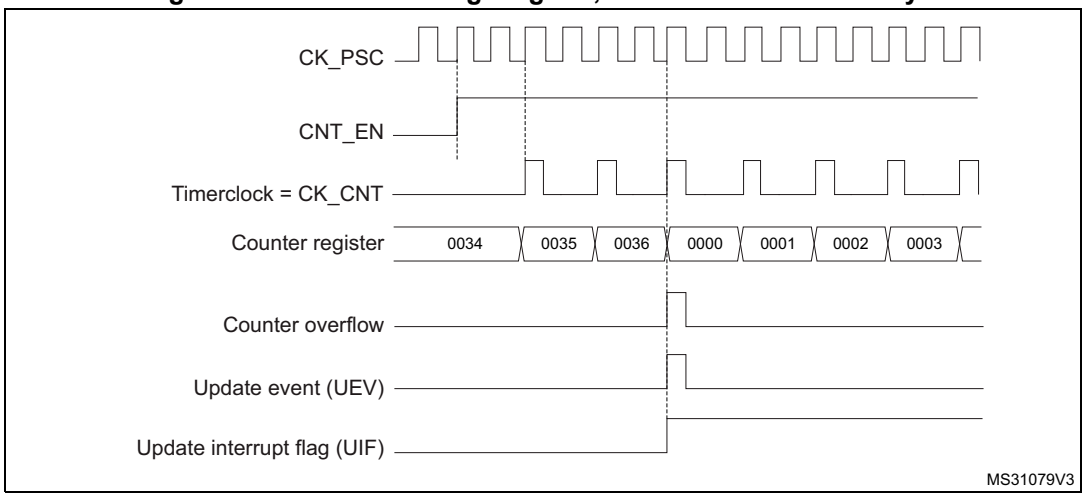


Figure 186. Counter timing diagram, internal clock divided by 4

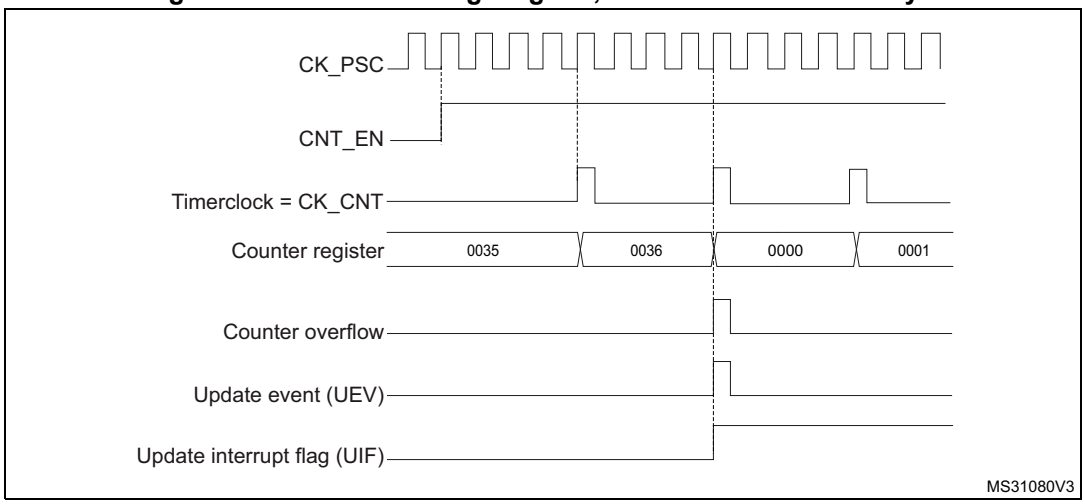


Figure 187. Counter timing diagram, internal clock divided by N

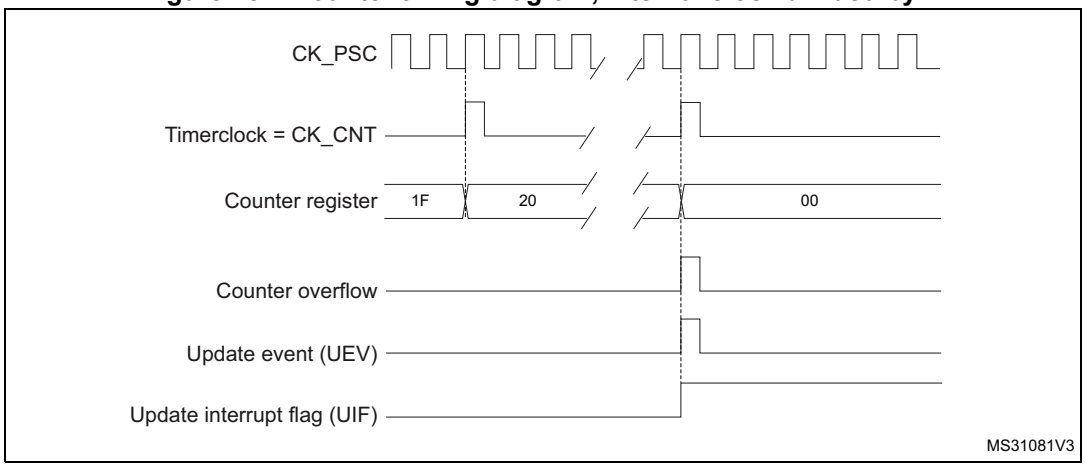


Figure 188. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

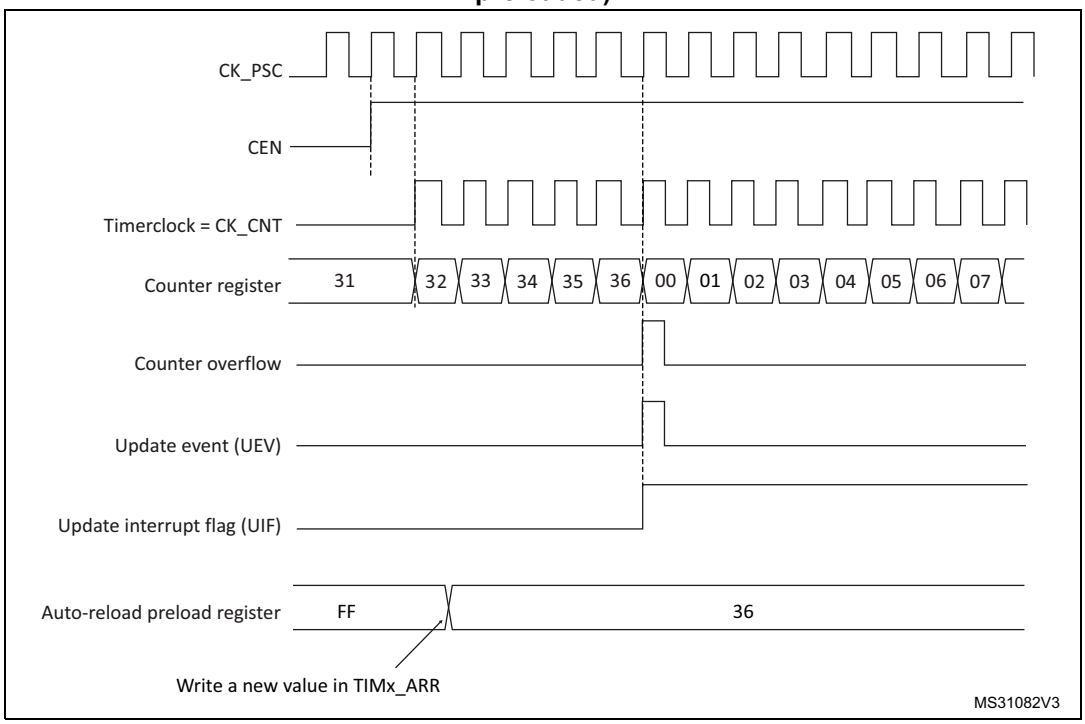
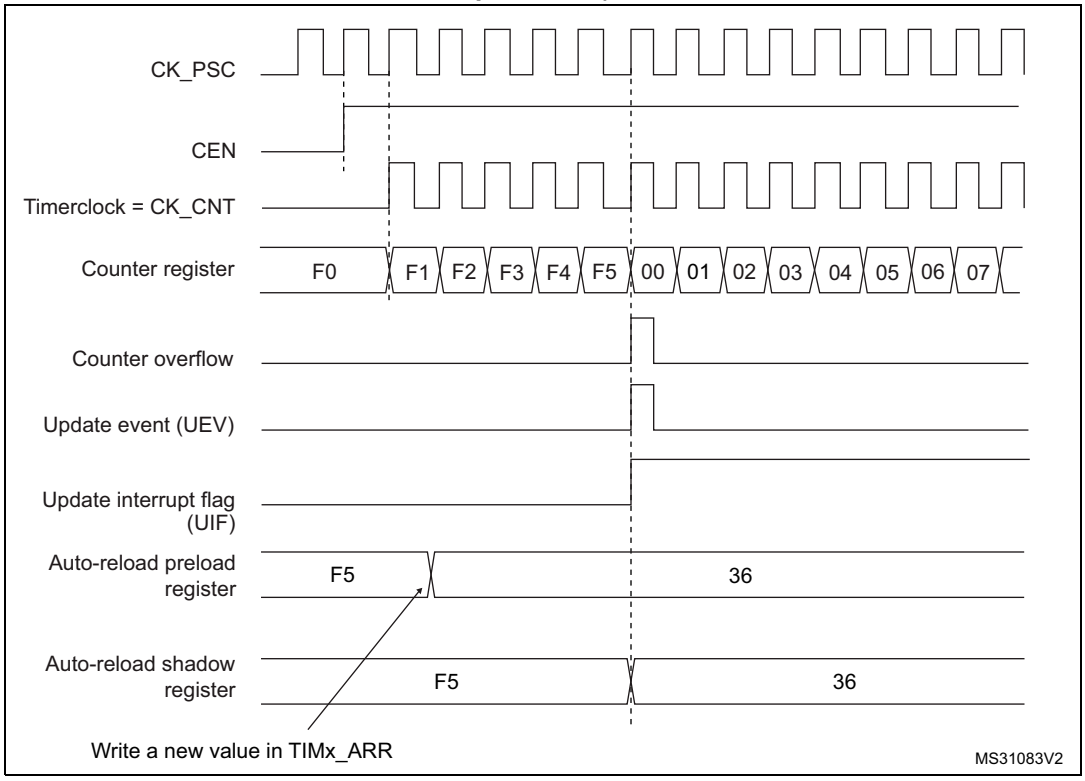


Figure 189. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



19.3.3 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1 (for **TIM9 and TIM1**2): external input pin (TIx)
- Internal trigger inputs (ITRx) (for **TIM9 and TIM1**2): connecting the trigger output from another timer. Refer to *Using one timer as prescaler for another timer* for more details.

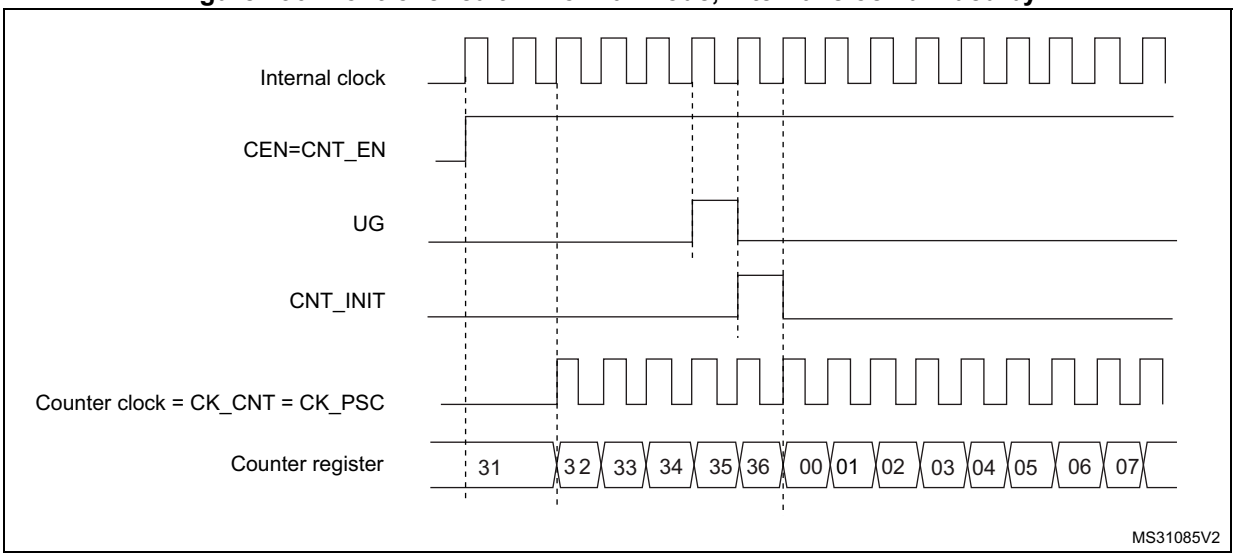
Internal clock source (CK_INT)

The internal clock source is the default clock source for TIM10/TIM11 and TIM13/TIM14.

For TIM9 and TIM12, the internal clock source is selected when the slave mode controller is disabled (SMS='000'). The CEN bit in the TIMx_CR1 register and the UG bit in the TIMx_EGR register are then used as control bits and can be changed only by software (except for UG which remains cleared). As soon as the CEN bit is programmed to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 190 shows the behavior of the control circuit and of the upcounter in normal mode, without prescaler.

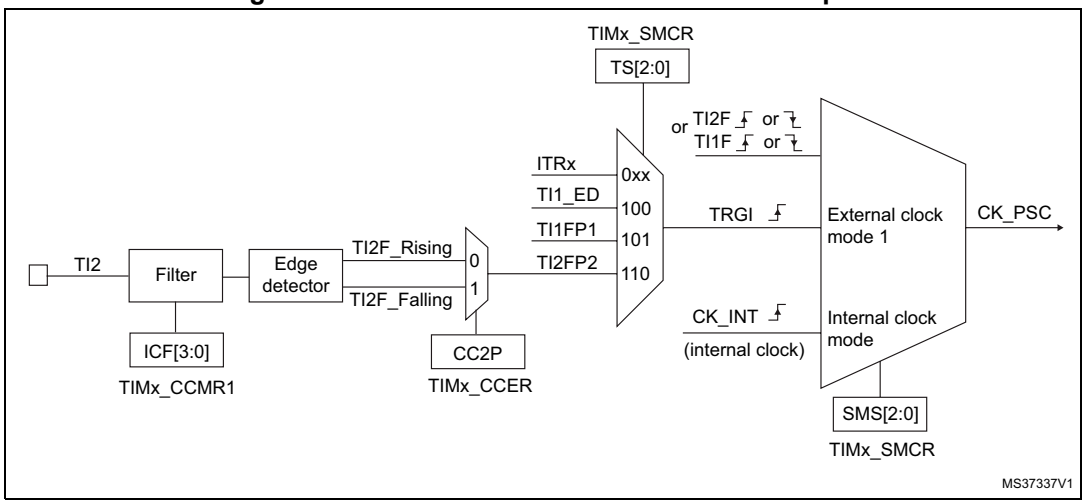
Figure 190. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1(TIM9 and TIM12)

This mode is selected when SMS='111' in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 191. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

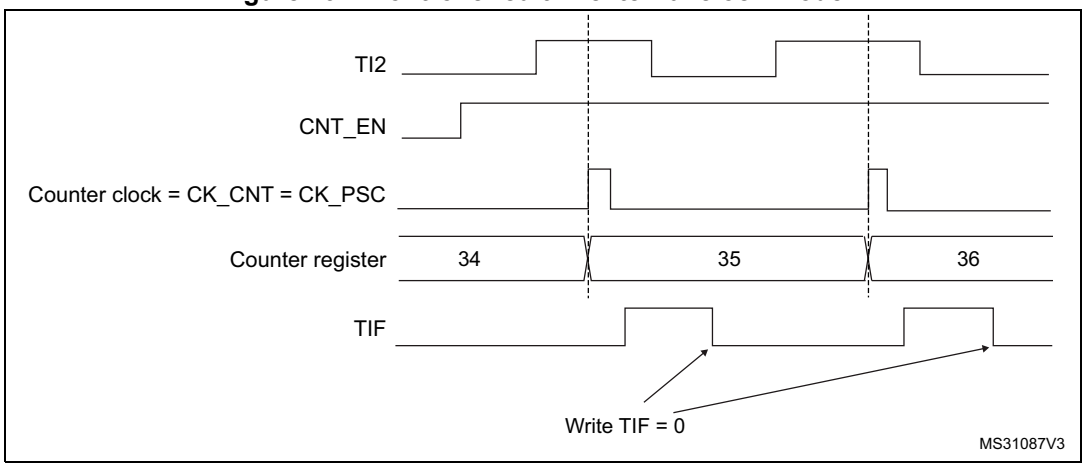
1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=’0000’).
3. Select the rising edge polarity by writing CC2P=’0’ and CC2NP=’0’ in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=’111’ in the TIMx_SMCR register.
5. Select TI2 as the trigger input source by writing TS=’110’ in the TIMx_SMCR register.
6. Enable the counter by writing CEN=’1’ in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so no 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 192. Control circuit in external clock mode 1



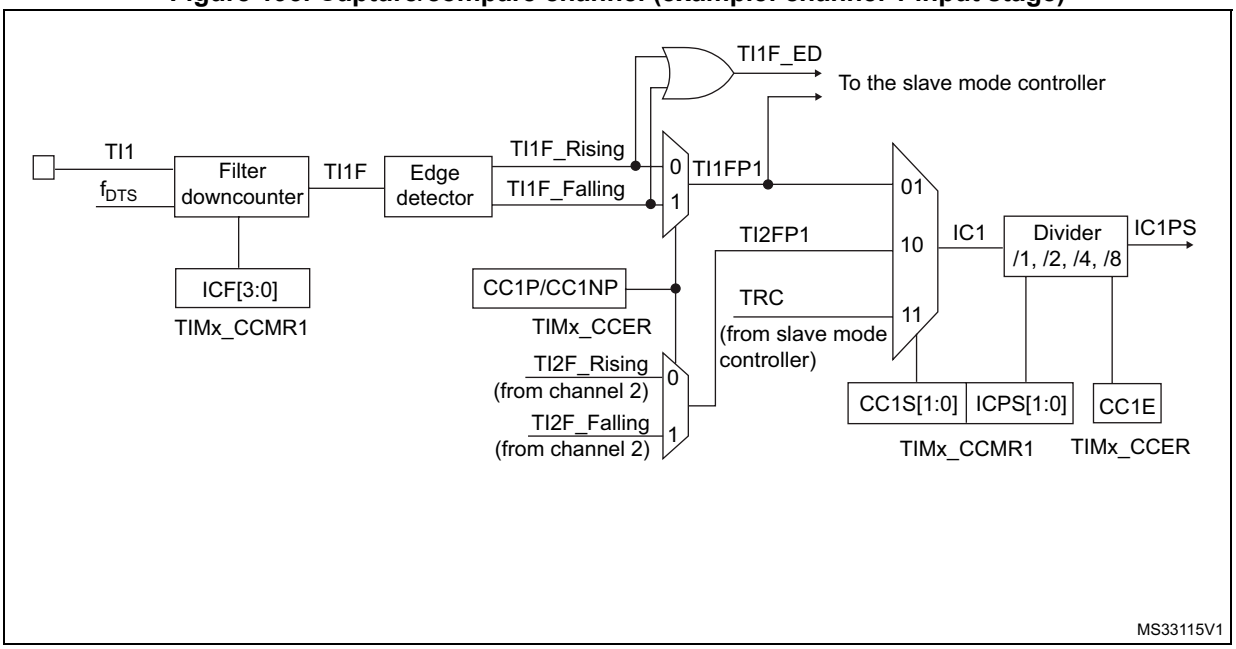
19.3.4 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 193 to *Figure 195* give an overview of a 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 193. 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 194. Capture/compare channel 1 main circuit

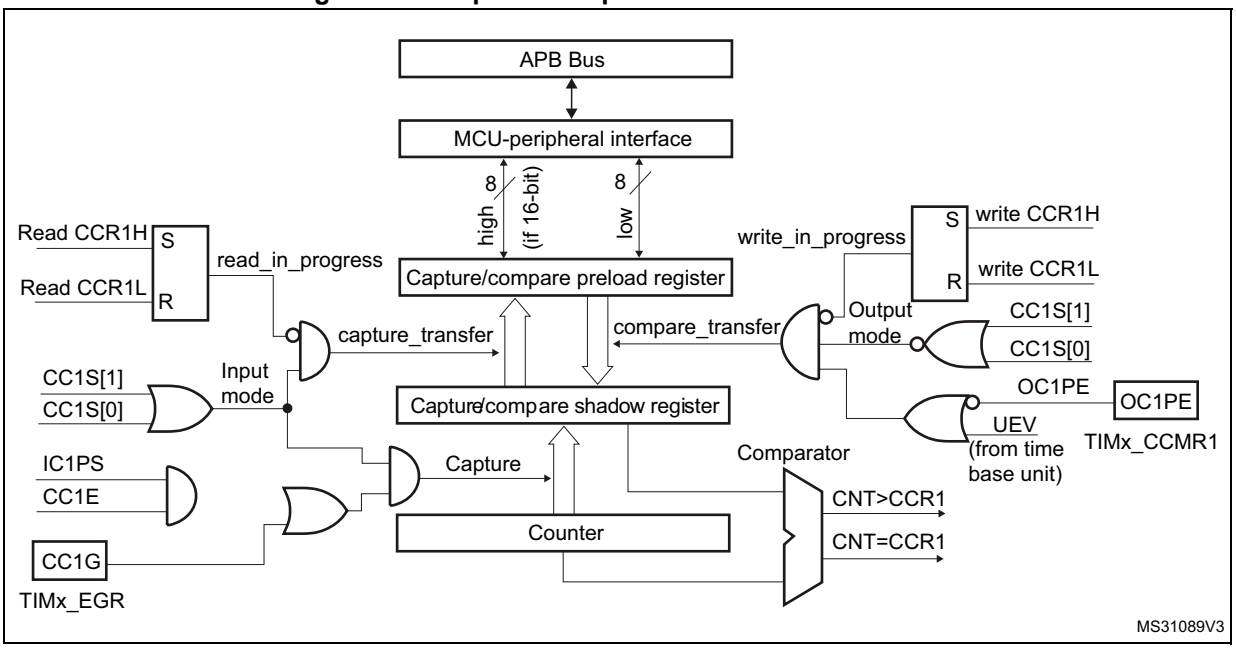
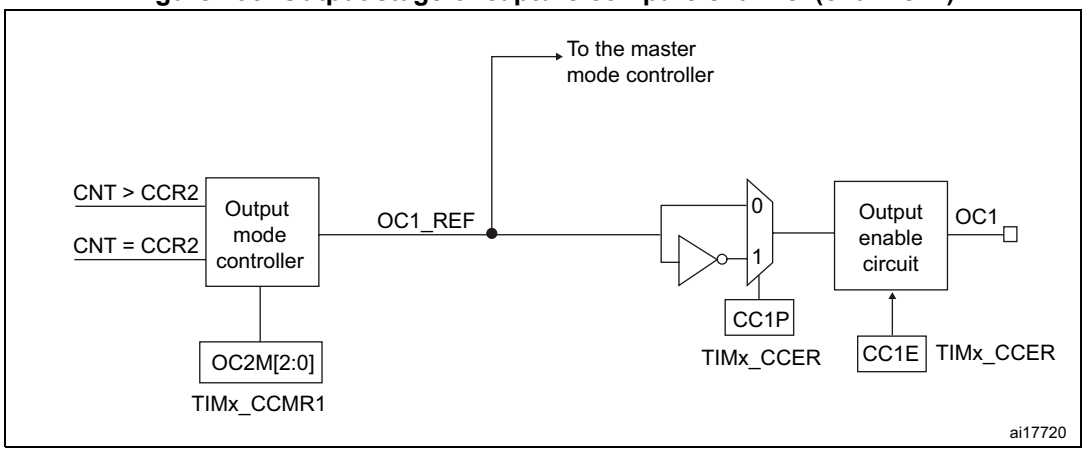


Figure 195. Output stage of capture/compare channel (channel 1)



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.

19.3.5 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 the user writes it to ‘0’.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to ‘01’ in the TIMx_CCMR1 register. As soon as CC1S becomes different from ‘00’, the channel is configured in input mode and the TIMx_CCR1 register becomes read-only.
2. Program the needed input filter duration with respect to the signal connected to the timer (by programming the ICxF bits in the TIMx_CCMRx register if the input is one of the TIx inputs). Let’s imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to ‘0011’ in the TIMx_CCMR1 register.
3. Select the edge of the active transition on the TI1 channel by programming CC1P and CC1NP bits to ‘00’ in the TIMx_CCER register (rising edge in this case).
4. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to ‘00’ in the TIMx_CCMR1 register).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
6. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

19.3.6 PWM input mode (only for TIM9/12)

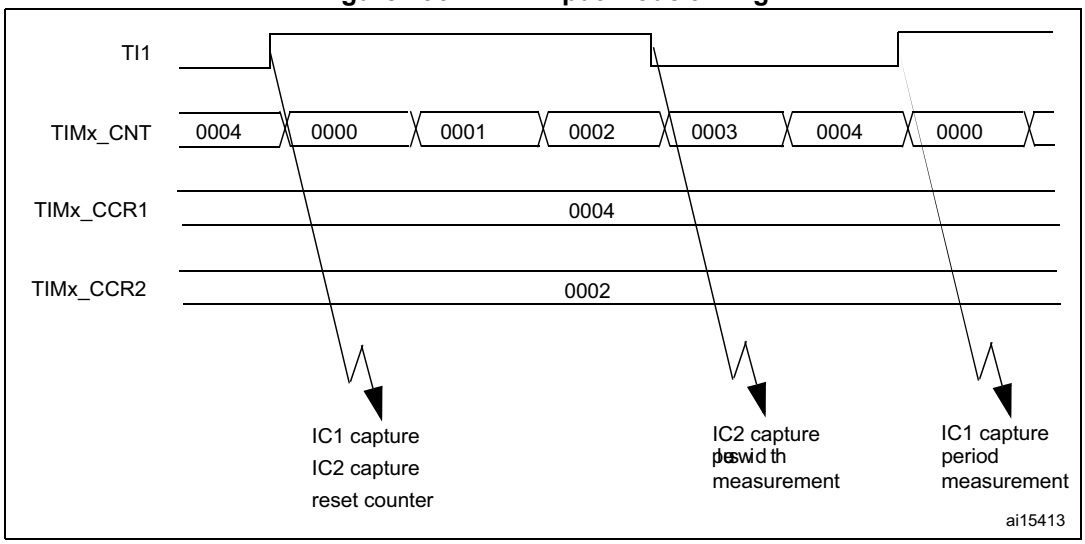
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, 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): program the CC1P and CC1NP bits to '00' (active on rising edge).
3. Select the active input for TIMx_CCR2: write the CC2S bits to '10' in the TIMx_CCMR1 register (TI1 selected).
4. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): program the CC2P and CC2NP bits to '11' (active on falling edge).
5. Select the valid trigger input: write the TS bits to '101' in the TIMx_SMCR register (TI1FP1 selected).
6. Configure the slave mode controller in reset mode: write the SMS bits to '100' in the TIMx_SMCR register.
7. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 196. 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.

19.3.7 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, the user just needs to write ‘101’ in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=’0’ (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to ‘100’ in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

19.3.8 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:

1. Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=’000’), be set active (OCxM=’001’), be set inactive (OCxM=’010’) or can toggle (OCxM=’011’) on match.
2. Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

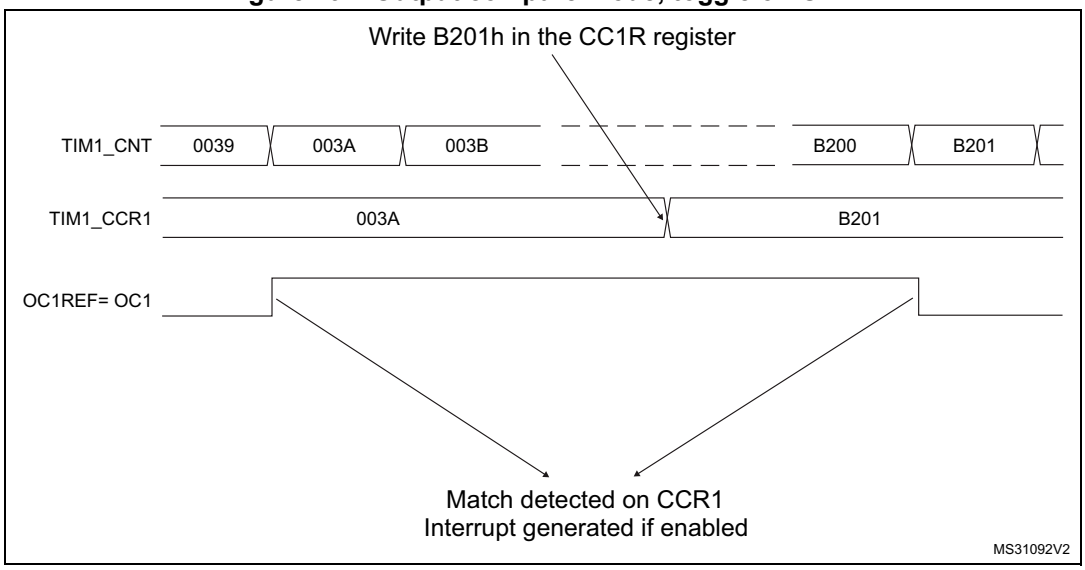
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = ‘011’ to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = ‘0’ to disable preload register
   - Write CCxP = ‘0’ to select active high polarity
   - Write CCxE = ‘1’ to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 197*.

Figure 197. Output compare mode, toggle on OC1.



19.3.9 PWM mode

Pulse Width Modulation mode allows the user 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. Enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register 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, the user has to initialize all the registers by setting the UG bit in the TIMx_EGR register.

The OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. The OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CNT ≤TIMx_CCRx.

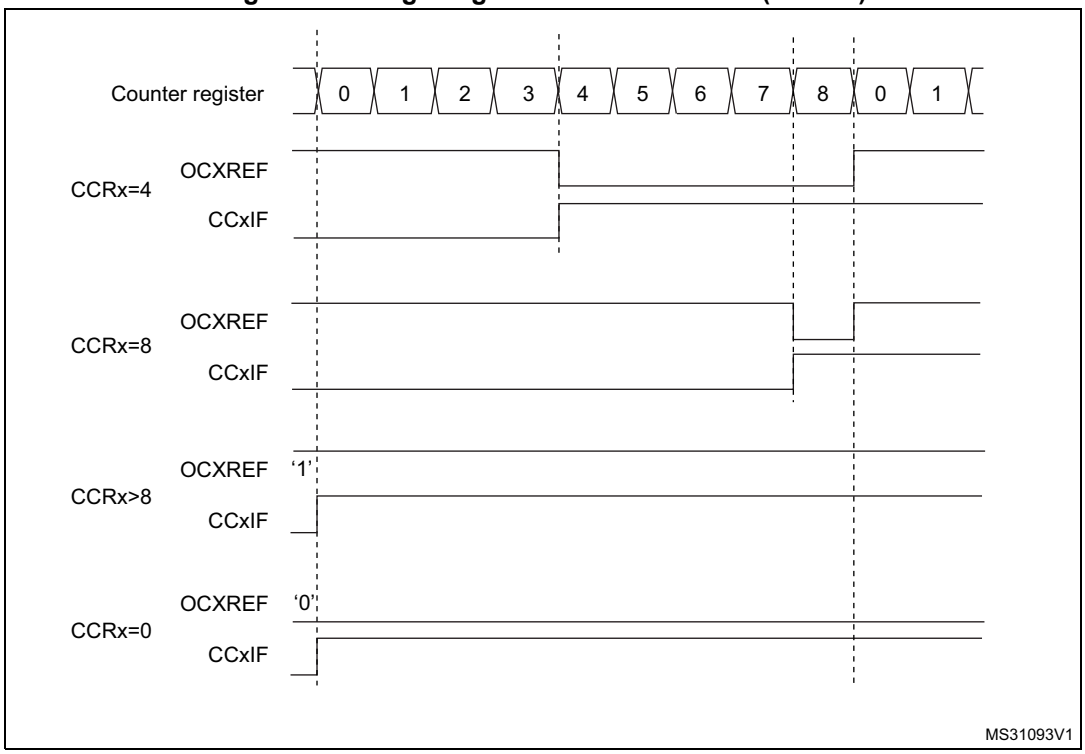
The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

PWM edge-aligned mode

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as 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 198* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 198. Edge-aligned PWM waveforms (ARR=8)



19.3.10 One-pulse mode

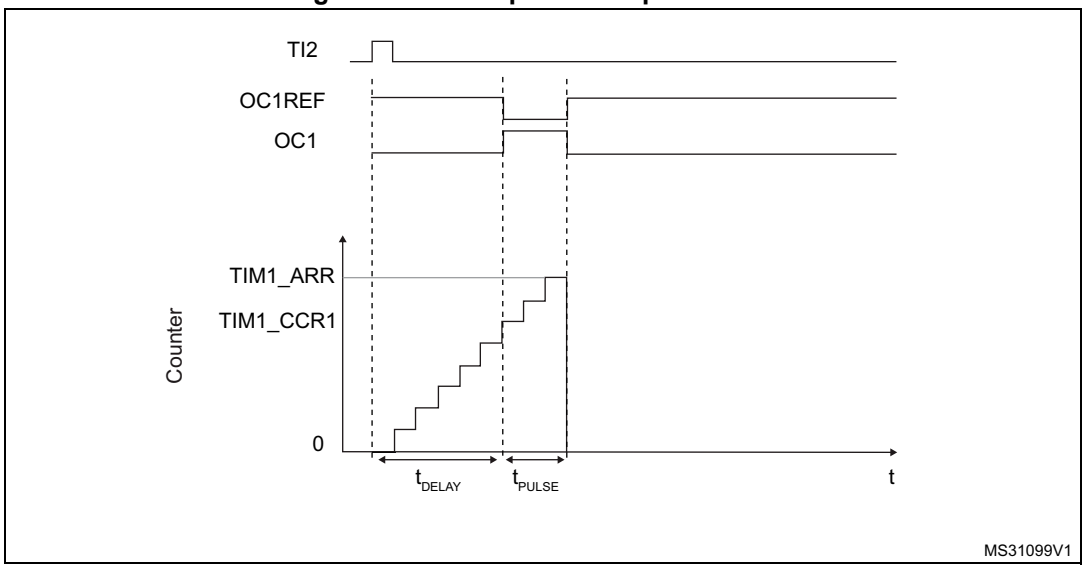
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. Select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be as follows:

CNT < CCRx≤ ARR (in particular, 0 < CCRx)

Figure 199. Example of one pulse mode.



For example the user may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Use TI2FP2 as trigger 1:

1. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
2. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP = '0' in the TIMx_CCER register.
3. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='110' in the TIMx_SMCR register.
4. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let us say the user wants 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 enable PWM mode 2 by writing OC1M='111' in the TIMx_CCMR1 register. The user 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 the user has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

The user only wants one pulse (Single mode), so write '1 in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If the user wants to output a waveform with the minimum delay, 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.

19.3.11 TIM9/12 external trigger synchronization

The TIM9/12 timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

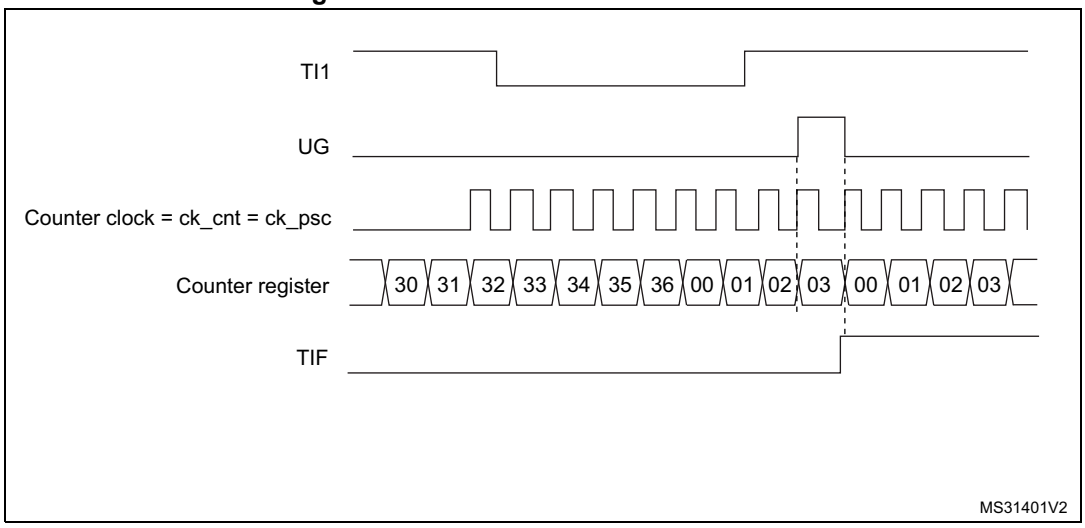
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F='0000'). The capture prescaler is not used for triggering, so there's no need to configure it. The CC1S bits select the input capture source only, CC1S = '01' in the TIMx_CCMR1 register. Program CC1P and CC1NP to '00' in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS='100' in TIMx_SMCR register. Select TI1 as the input source by writing TS='101' in TIMx_SMCR register.
3. Start the counter by writing CEN='1' in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request can be sent if enabled (depending on the TIE bit in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 200. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

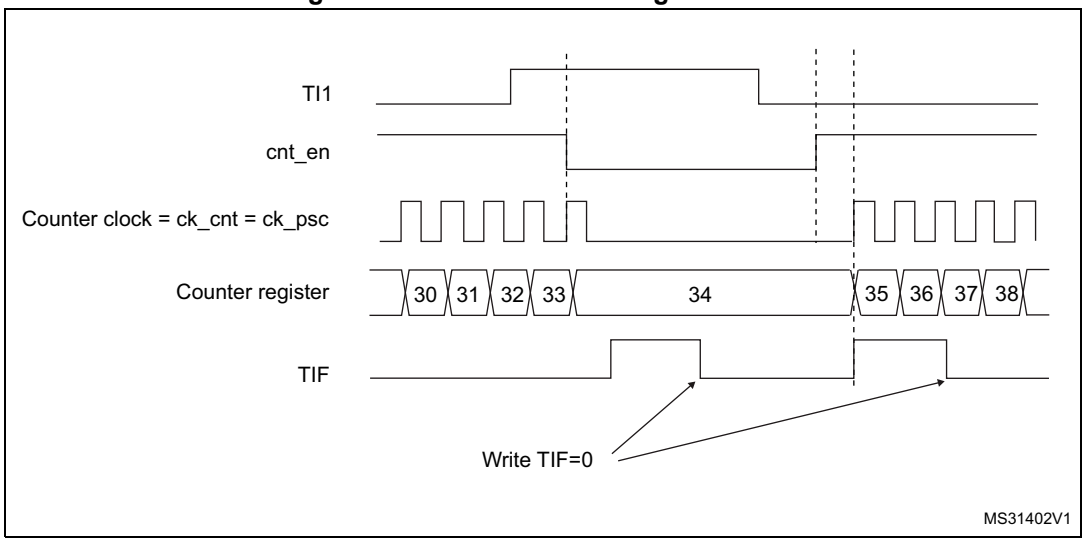
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F='0000'). The capture prescaler is not used for triggering, so there's no need to configure it. The CC1S bits select the input capture source only, CC1S='01' in TIMx_CCMR1 register. Program CC1P='1' and CC1NP= '0' in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS='101' in TIMx_SMCR register. Select TI1 as the input source by writing TS='101' in TIMx_SMCR register.
3. Enable the counter by writing CEN='1' in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN='0', whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 201. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

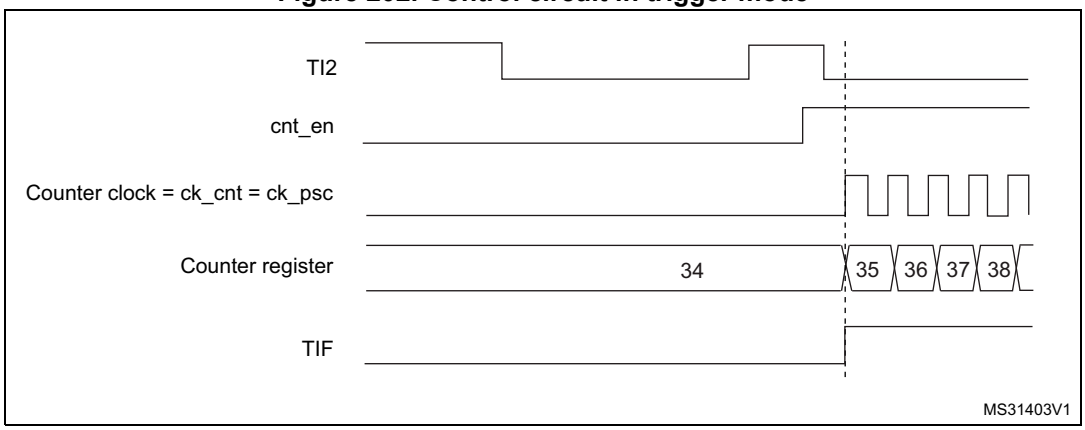
In the following example, the upcounter starts in response to a rising edge on TI2 input:

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 there's no need to configure it. The CC2S bits are configured to select the input capture source only, CC2S='01' in TIMx_CCMR1 register. Program CC2P='1' and CC2NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS='110' in TIMx_SMCR register. Select TI2 as the input source by writing TS='110' in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 202. Control circuit in trigger mode



19.3.12 Timer synchronization (TIM9/12)

The TIM timers are linked together internally for timer synchronization or chaining. Refer to *Section 18.3.15: Timer synchronization* for details.

Note: The clock of the slave timer 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.

19.3.13 Debug mode

When the microcontroller enters debug mode (Cortex®-M4 with FPU 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 38.16.2: Debug support for timers, watchdog, bxCAN and I^2C*.

19.4 TIM9 and TIM12 registers

Refer to *Section 1.1* for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

19.4.1 TIM9/12 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
						rw	rw	rw				rw	rw	rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **CKD**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (TIx),

00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 \times t_{CK\_INT}$
10: $t_{DTS} = 4 \times t_{CK\_INT}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generates an update interrupt if enabled:
- Counter overflow
- Setting the UG bit

1: Only counter overflow generates an update interrupt if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable update event (UEV) generation.
0: UEV enabled. An UEV is generated by one of the following events:
- Counter overflow
- Setting the UG bit

Buffered registers are then loaded with their preload values.
1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

CEN is cleared automatically in one-pulse mode, when an update event occurs.

19.4.2 TIM9/12 slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

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

Bit 7 **MSM:** Master/Slave mode

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful in order to synchronize several timers on a single external event.

Bits 6:4 **TS:** Trigger selection

This 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: Reserved.

See *Table 101* for more details on the meaning of ITRx for each timer.

Note: These bits must be changed only when they are not used (e.g. when SMS='000') to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SMS:** Slave mode selection

When external signals are selected, the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input control register and Control register descriptions.
000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock
001: Reserved
010: Reserved
011: Reserved
100: Reset mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers
101: Gated mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Counter starts and stops are both controlled
110: Trigger mode - The counter starts on a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled
111: External clock mode 1 - Rising edges of the selected trigger (TRGI) clock the counter

Note: The Gated mode must not be used if TI1F_ED is selected as the trigger input (TS='100'). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the Gated mode checks the level of the trigger signal.

Note: The clock of the slave timer 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.

Table 101. TIMx internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM9	TIM2_TRGO	TIM3_TRGO	TIM10_OC	TIM11_OC
TIM12	TIM4_TRGO	TIM5_TRGO	TIM13_OC	TIM14_OC

19.4.3 TIM9/12 Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TIE	Res			CC2IE	CC1IE	UIE
									rw				rw	rw	rw

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

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

19.4.4 TIM9/12 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CC2OF	CC1OF	Reserved		TIF	Reserved			CC2IF	CC1IF	UIF
					rc_w0	rc_w0			rc_w0				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

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.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
If channel CC1 is configured as output:
This flag is set by hardware when the counter matches the compare value. It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow.
If channel CC1 is configured as input:
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred.
1: The counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1 which matches the selected polarity).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow 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.

19.4.5 TIM9/12 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TG	Reserved			CC2G	CC1G	UG
									w				w	w	w

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

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in the TIMx_SR register. Related interrupt can occur if enabled

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/compare 2 generation

refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

the CC1IF flag is set, the corresponding interrupt is sent if enabled.

If channel CC1 is configured as input:

The current counter value is captured in the TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Re-initializes the counter and generates an update of the registers. The prescaler counter is also cleared and the prescaler ratio is not affected. The counter is cleared.

19.4.6 TIM9/12 capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes. For a given bit, OCxx describes its function when the channel is configured in output mode, ICxx describes its function when the channel is configured in input mode. Take care that the same bit can have different meanings for the input stage and the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **OC2M[2:0]**: Output compare 2 mode

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register

Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **OC1M**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas the active levels of OC1 and OC1N depend on the CC1P and CC1NP bits, respectively.
000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).
001: Set channel 1 to active level on match. The OC1REF signal is forced high when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).
010: Set channel 1 to inactive level on match. The OC1REF signal is forced low when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).
011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1
100: Force inactive level - OC1REF is forced low
101: Force active level - OC1REF is forced high
110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else it is inactive. In downcounting, channel 1 is inactive (OC1REF=‘0) as long as TIMx_CNT>TIMx_CCR1, else it is active (OC1REF=’1’)
111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else it is active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else it is inactive.

Note: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken into account immediately
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded into the active register at each update event

Note: The PWM mode can be used without validating the preload register only in one-pulse mode (OPM bit set in the TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on the counter and CCR1 values even when the trigger is ON. The minimum delay to activate the CC1 output when an edge occurs on the trigger input is 5 clock cycles
1: An active edge on the trigger input acts like a compare match on the CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

Input capture mode

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F**: Input capture 1 filter

This bitfield defines the frequency used to sample the TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N 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 bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

19.4.7 TIM9/12 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

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

Bit 7 **CC2NP**: Capture/Compare 2 output Polarity
refer to CC1NP description

Bits 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output Polarity
refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity
CC1 channel configured as output: CC1NP must be kept cleared
CC1 channel configured as input: CC1NP is used in conjunction with CC1P to define TI1FP1/TI2FP1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.
CC1 channel configured as output:
0: OC1 active high.
1: OC1 active low.
CC1 channel configured as input:
CC1NP/CC1P bits select TI1FP1 and TI2FP1 polarity for trigger or capture operations.
00: noninverted/rising edge
Circuit is sensitive to TIxFP1 rising edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode, encoder mode).
01: inverted/falling edge
Circuit is sensitive to TIxFP1 falling edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is inverted (trigger in gated mode, encoder mode).
10: reserved, do not use this configuration.

Note: 11: noninverted/both edges
Circuit is sensitive to both TIxFP1 rising and falling edges (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode). This configuration must not be used for encoder mode.

Bit 0 **CC1E**: Capture/Compare 1 output enable.
CC1 channel configured as output:
0: Off - OC1 is not active.
1: On - OC1 signal is output on the corresponding output pin.
CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled.
1: Capture enabled.

Table 102. 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 states of the external I/O pins connected to the standard OCx channels depend on the state of the OCx channel and on the GPIO registers.

19.4.8 TIM9/12 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

Bits 15:0 **CNT[15:0]**: Counter value

19.4.9 TIM9/12 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded 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").

19.4.10 TIM9/12 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 into the actual auto-reload register.
Refer to the *Section 19.3.1: Time-base unit* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

19.4.11 TIM9/12 capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded into the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (OC1PE bit). Else the preload value is copied into the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the TIMx_CNT counter and signaled on the OC1 output.

If channel CC1is 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.

19.4.12 TIM9/12 capture/compare register 2 (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded into the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (OC2PE bit). Else the preload value is copied into the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the TIMx_CNT counter and signalled on the OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

19.4.13 TIM9/12 register map

TIM9/12 registers are mapped as 16-bit addressable registers as described below. The reserved memory areas are highlighted in gray in the table.

Table 103. TIM9/12 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Reserved																						CKD [1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value																							0	0	0				0	0	0	0
0x08	**TIMx_SMCR**	Reserved																								MSM	TS[2:0]			Reserved	SMS[2:0]		
	Reset value																									0	0	0	0		0	0	0
0x0C	**TIMx_DIER**	Reserved																									TIE	Reserved			CC2IE	CC1IE	UIE
	Reset value																										0				0	0	0
0x10	**TIMx_SR**	Reserved																						CC2OF	CC1OF	Reserved	TIF	Reserved			CC2IF	CC1IF	UIF
	Reset value																							0	0		0				0	0	0
0x14	**TIMx_EGR**	Reserved																									TG	Reserved			CC2G	CC1G	UG
	Reset value																										0				0	0	0
0x18	**TIMx_CCMR1** Output Compare mode	Reserved																	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Reserved	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																		0	0	0	0	0	0	0		0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Reserved																IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	Reserved																																
0x20	**TIMx_CCER**	Reserved																								CC2NP	Reserved	CC2P	CC2E	CC1NP	Reserved	CC1P	CC1E
	Reset value																									0		0	0	0		0	0
0x24	**TIMx_CNT**	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved																																
0x34	**TIMx_CCR1**	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 103. TIM9/12 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
0x38	**TIMx_CCR2**	Reserved																CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C to 0x4C	Reserved																																

Refer to *Section 2.3: Memory map* for the register boundary addresses.

19.5 TIM10/11/13/14 registers

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

19.5.1 TIM10/11/13/14 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
						rw	rw	rw				rw	rw	rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **CKD**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (TIx),

00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 \times t_{CK\_INT}$
10: $t_{DTS} = 4 \times t_{CK\_INT}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the update interrupt (UEV) sources.

0: Any of the following events generate an UEV if enabled:
- Counter overflow
- Setting the UG bit

1: Only counter overflow generates an UEV if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable update interrupt (UEV) event generation.

0: UEV enabled. An UEV is generated by one of the following events:
- Counter overflow
- Setting the UG bit.

Buffered registers are then loaded with their preload values.

1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

19.5.2 TIM10/11/13/14 Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1IE	UIE
														rw	rw

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

19.5.3 TIM10/11/13/14 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CC1OF	Reserved							CC1IF	UIF
						rc_w0								rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
If channel CC1 is configured as output:
This flag is set by hardware when the counter matches the compare value. It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow.
If channel CC1 is configured as input:
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred.
1: The counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1 which matches the selected polarity).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow and if UDIS='0' in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS='0' and UDIS='0' in the TIMx_CR1 register.

19.5.4 TIM10/11/13/14 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1G	UG
														w	w

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared.

19.5.5 TIM10/11/13/14 capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
Reserved								IC1F[3:0]				IC1PSC[1:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

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

Bits 6:4 **OC1M**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 is derived. OC1REF is active high whereas OC1 active level depends on CC1P bit.

000: Frozen. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

011: Toggle - OC1REF toggles when TIMx_CNT = TIMx_CCR1.

100: Force inactive level - OC1REF is forced low.

101: Force active level - OC1REF is forced high.

110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT < TIMx_CCR1 else inactive.

111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT < TIMx_CCR1 else active.

Note: In PWM mode 1 or 2, the OCREF level changes when the result of the comparison changes or when the output compare mode switches from frozen to PWM mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger 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. OC is then set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, IC1 is mapped on TI1.

10:

11:

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

Input capture mode

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

Bits 7:4 **IC1F**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N 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: Reserved
11: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

19.5.6 TIM10/11/13/14 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CC1NP	Res.	CC1P	CC1E
												rw		rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity.
CC1 channel configured as output: CC1NP must be kept cleared.
CC1 channel configured as input: CC1NP bit is used in conjunction with CC1P to define TI1FP1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.
CC1 channel configured as output:
0: OC1 active high
1: OC1 active low
CC1 channel configured as input:
The CC1P bit selects TI1FP1 and TI2FP1 polarity for trigger or capture operations.
00: noninverted/rising edge
Circuit is sensitive to TI1FP1 rising edge (capture mode), TI1FP1 is not inverted.
01: inverted/falling edge
Circuit is sensitive to TI1FP1 falling edge (capture mode), TI1FP1 is inverted.
10: reserved, do not use this configuration.
11: noninverted/both edges
Circuit is sensitive to both TI1FP1 rising and falling edges (capture mode), TI1FP1 is not inverted.

Bit 0 **CC1E**: Capture/Compare 1 output enable.
CC1 channel configured as output:
0: Off - OC1 is not active
1: On - OC1 signal is output on the corresponding output pin
CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled
1: Capture enabled

Table 104. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output Disabled (OCx='0', OCx_EN='0')
1	OCx=OCxREF + Polarity, OCx_EN='1'

Note: The state of the external I/O pins connected to the standard OCx channels depends on the OCx channel state and the GPIO registers.

19.5.7 TIM10/11/13/14 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

Bits 15:0 **CNT[15:0]**: Counter value

19.5.8 TIM10/11/13/14 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

19.5.9 TIM10/11/13/14 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 *Section 19.3.1: Time-base unit* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

19.5.10 TIM10/11/13/14 capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

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 CC1is 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.

19.5.11 TIM11 option register 1 (TIM11_OR)

Address offset: 0x50

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														TI1_RMP[1:0]	
														rw	

Bits 15:2 Reserved, must be kept at reset value.

Bits 1:0 **TI1_RMP[1:0]**: TIM11 Input 1 remapping capability

Set and cleared by software.

00,01,11: TIM11 Channel1 is connected to the GPIO (refer to the Alternate function mapping table in the datasheets).

10: HSE_RTC clock (HSE divided by programmable prescaler) is connected to the TIM11_CH1 input for measurement purposes

19.5.12 TIM10/11/13/14 register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below.

Table 105. TIM10/11/13/14 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Reserved																						CKD [1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value																							0	0	0				0	0	0	0
0x08	**TIMx_SMCR**	Reserved																															
	Reset value																																
0x0C	**TIMx_DIER**	Reserved																														CC1IE	UIE
	Reset value																															0	0
0x10	**TIMx_SR**	Reserved																						CC1OF	Reserved							CC1IF	UIF
	Reset value																							0								0	0
0x14	**TIMx_EGR**	Reserved																														CC1G	UG
	Reset value																															0	0
0x18	**TIMx_CCMR1** Output compare mode	Reserved																									OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																										0	0	0	0	0	0	0
	TIMx_CCMR1 Input capture mode	Reserved																									IC1F[3:0]				IC1 PSC [1:0]	CC1S [1:0]	
	Reset value																										0	0	0	0	0	0	0
0x1C	Reserved																																
0x20	**TIMx_CCER**	Reserved																													CC1NP	Reserved	CC1P	CC1E
	Reset value																													0		0	0
0x24	**TIMx_CNT**	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved																																
0x34	**TIMx_CCR1**	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 to 0x4C	Reserved																																

Table 105. TIM10/11/13/14 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x50	TIMx_OR	Reserved																														TI1_RMP	
	Reset value																															0	0

Refer to *Section 2.3: Memory map* for the register boundary addresses.

20 Basic timers (TIM6 and TIM7)

This section applies to the whole STM32F4xx family, unless otherwise specified.

20.1 TIM6 and 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.

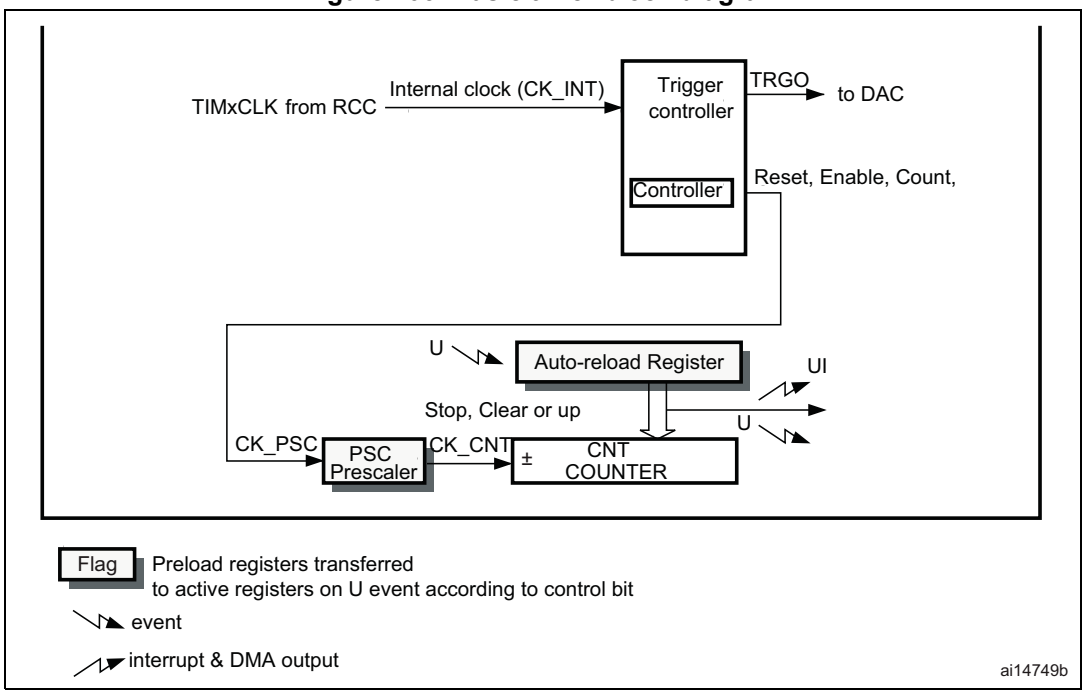
The timers are completely independent, and do not share any resources.

20.2 TIM6 and TIM7 main features

Basic timer (TIM6 and 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 65536
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

Figure 203. Basic timer block diagram



20.3 TIM6 and TIM7 functional description

20.3.1 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 204 and *Figure 205* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 204. Counter timing diagram with prescaler division change from 1 to 2

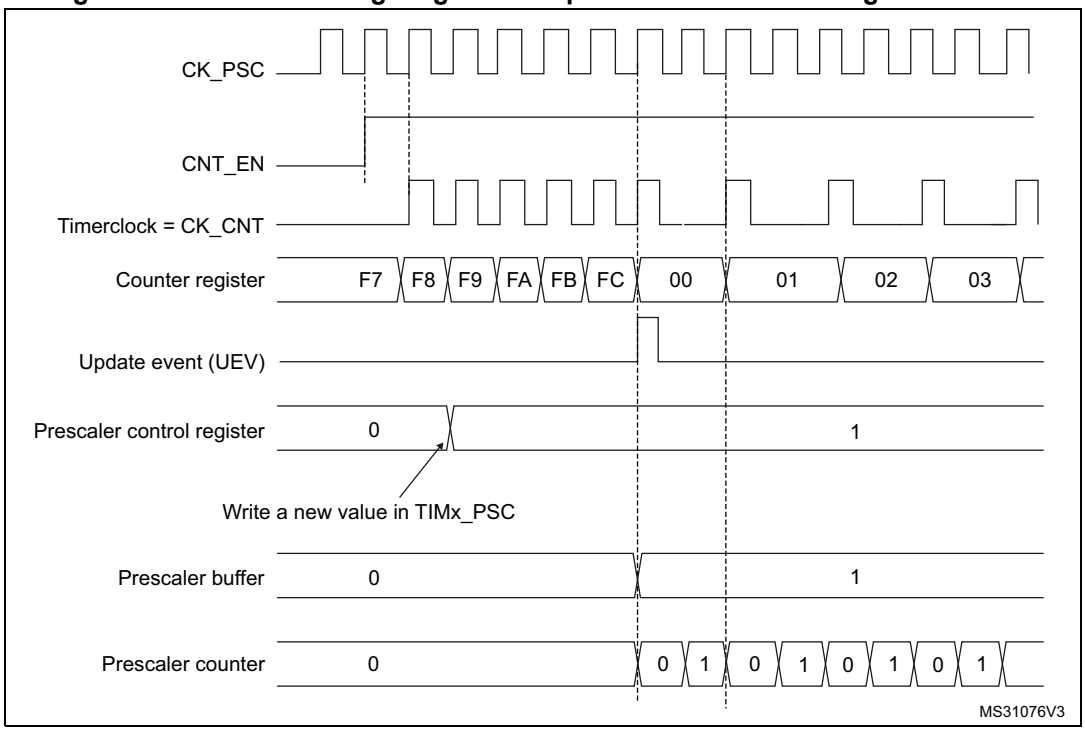
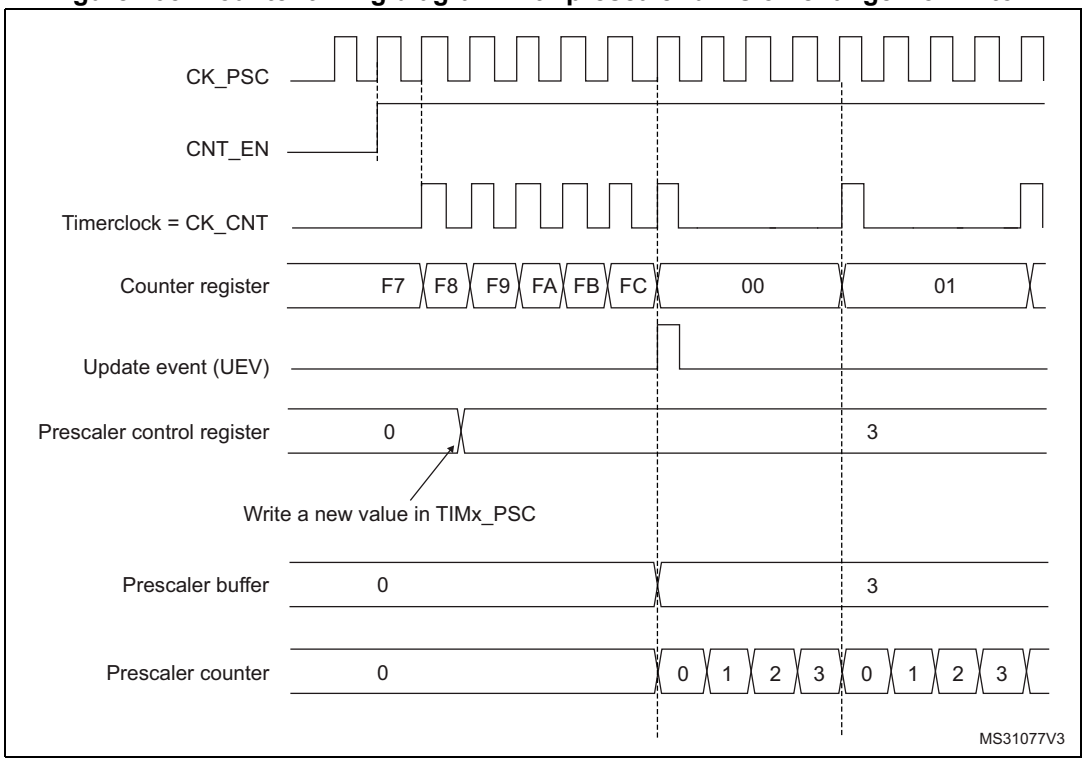


Figure 205. Counter timing diagram with prescaler division change from 1 to 4



20.3.2 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 generate 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 206. Counter timing diagram, internal clock divided by 1

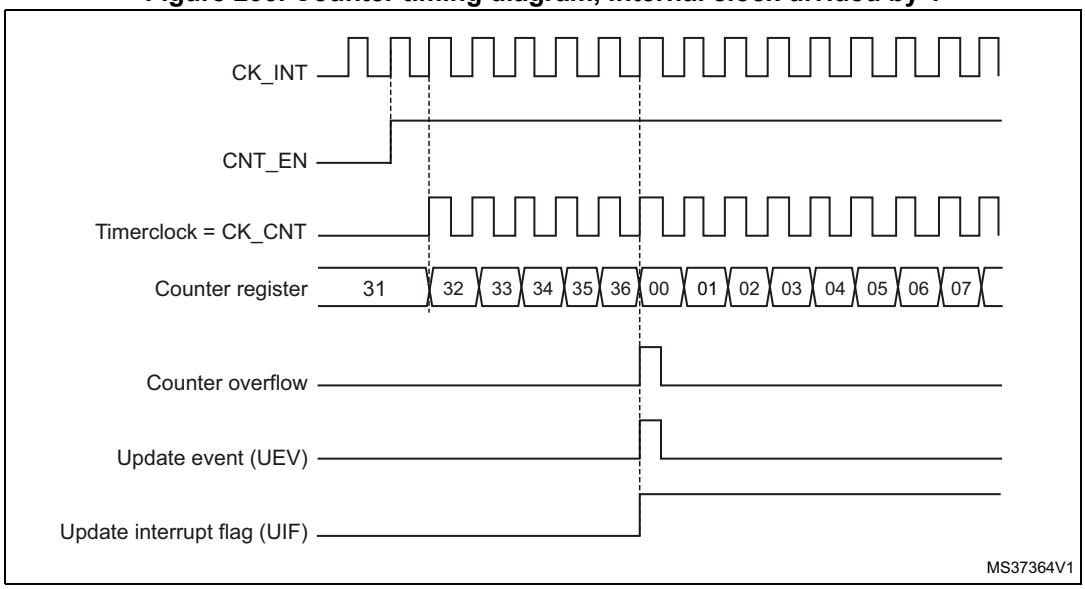


Figure 207. Counter timing diagram, internal clock divided by 2

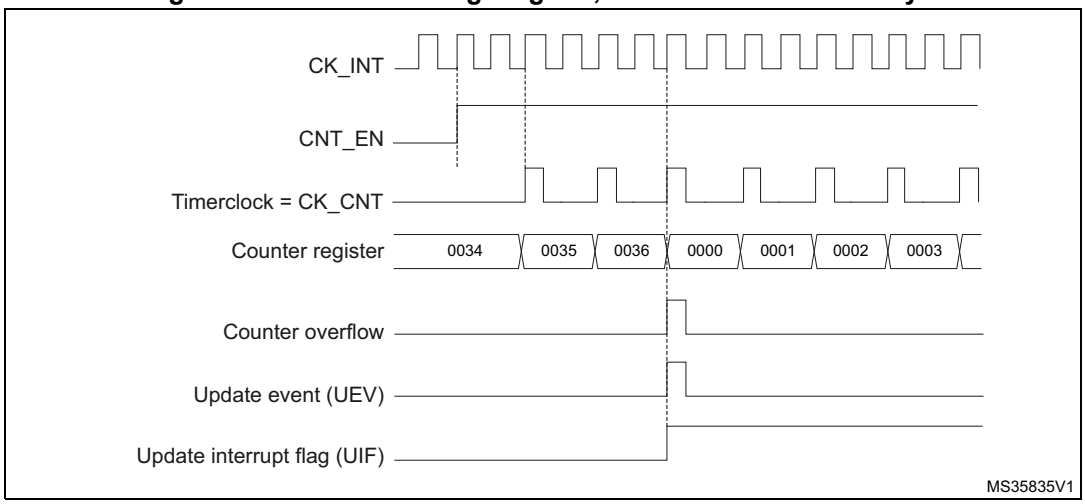


Figure 208. Counter timing diagram, internal clock divided by 4

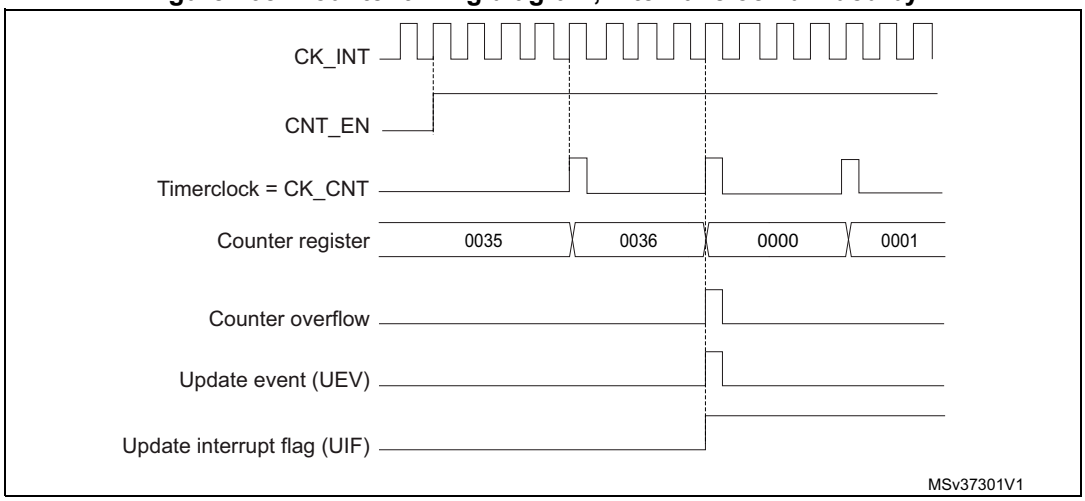


Figure 209. Counter timing diagram, internal clock divided by N

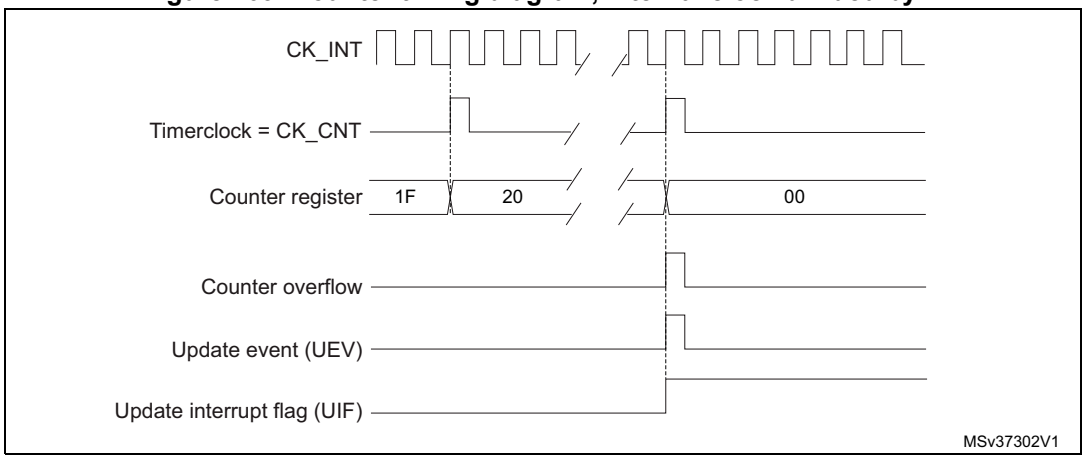


Figure 210. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

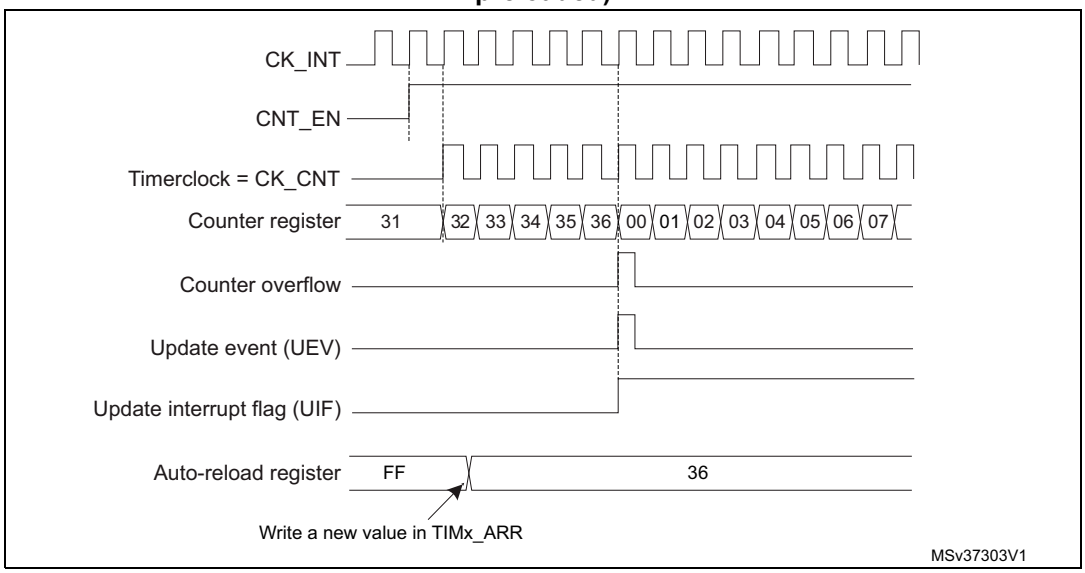
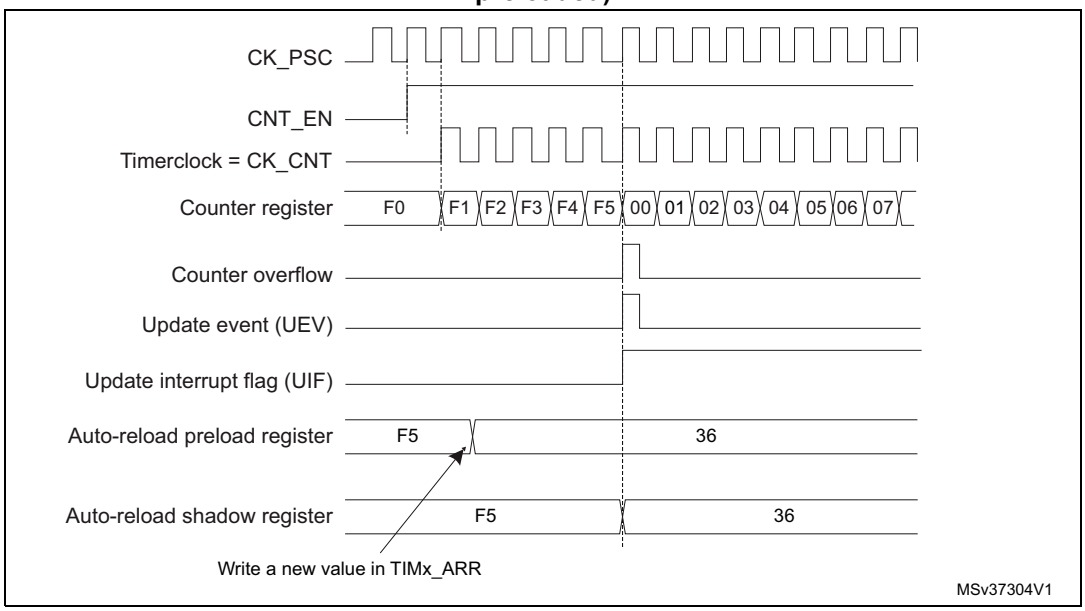


Figure 211. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



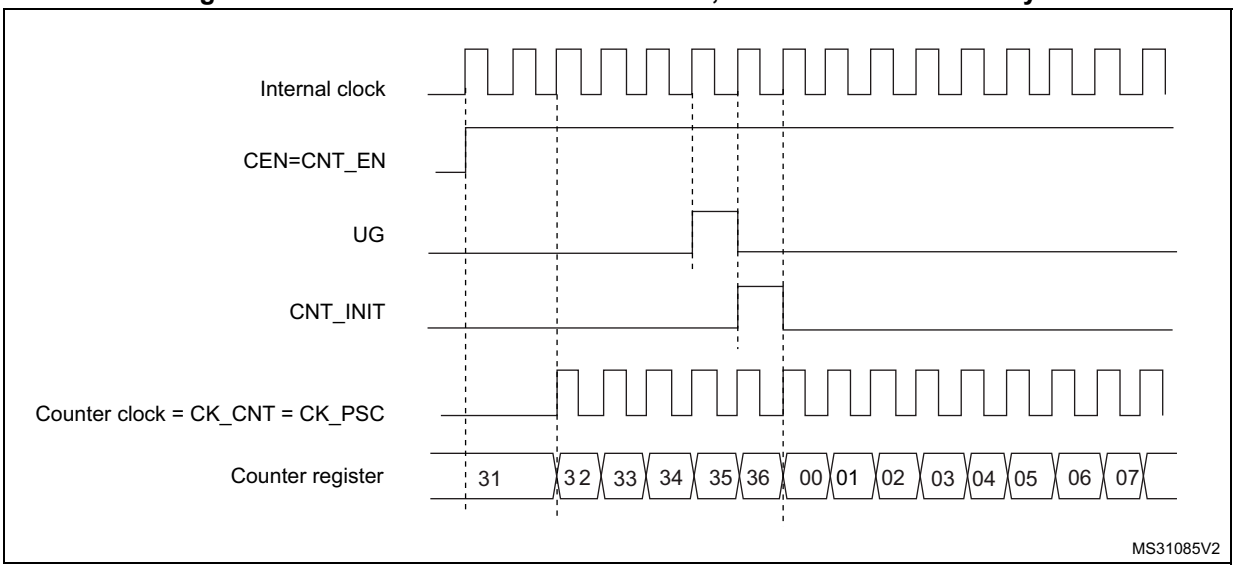
20.3.3 Clock source

The counter clock is provided by the Internal clock (CK_INT) source.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 212 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 212. Control circuit in normal mode, internal clock divided by 1



20.3.4 Debug mode

When the microcontroller enters the debug mode (Cortex®-M4 with FPU 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 38.16.2: Debug support for timers, watchdog, bxCAN and I^2C*.

20.4 TIM6 and TIM7 registers

Refer to *Section 1.1* for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

20.4.1 TIM6 and 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
Reserved								ARPE	Reserved			OPM	URS	UDIS	CEN
								rw				rw	rw	rw	rw

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

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: Gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

20.4.2 TIM6 and TIM7 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									MMS[2:0]			Reserved			
									rw	rw	rw				

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

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx_EGR register is used as a trigger output (TRGO). If reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter enable signal, CNT_EN, is used as a trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIMx_SMCR register).

010: **Update** - The update event is selected as a trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

Note: The clock of the slave timer and ADC must be enabled prior to receiving 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.

20.4.3 TIM6 and TIM7 DMA/Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							UDE	Reserved							UIE
							rw								rw

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.

20.4.4 TIM6 and TIM7 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow or underflow 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.

20.4.5 TIM6 and TIM7 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															UG
															w

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

20.4.6 TIM6 and TIM7 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

Bits 15:0 **CNT[15:0]**: Counter value

20.4.7 TIM6 and TIM7 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

20.4.8 TIM6 and TIM7 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 into the actual auto-reload register.
Refer to *Section 20.3.1: Time-base unit* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

20.4.9 TIM6 and TIM7 register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below.

Table 106. TIM6 and TIM7 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																								ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value																									0				0	0	0	0
0x04	TIMx_CR2	Reserved																									MMS[2:0]			Reserved			
	Reset value																										0	0	0				
0x08	Reserved																																
0x0C	TIMx_DIER	Reserved																								UDE	Reserved						UIE
	Reset value																									0							0
0x10	TIMx_SR	Reserved																															UIF
	Reset value																																0
0x14	TIMx_EGR	Reserved																															UG
	Reset value																																0
0x18	Reserved																																
0x1C	Reserved																																
0x20	Reserved																																
0x24	TIMx_CNT	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Reserved																ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.3: Memory map* for the register boundary addresses.

21 Independent watchdog (IWDG)

This section applies to the whole STM32F4xx family, unless otherwise specified.

21.1 IWDG introduction

The devices have two embedded watchdog peripherals which offer a combination of high safety level, timing accuracy and flexibility of use. Both watchdog peripherals (Independent and Window) serve to detect and resolve malfunctions due to software failure, and to trigger system reset or an interrupt (window watchdog only) when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails. The window watchdog (WWDG) clock is prescaled from the APB1 clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The IWDG is best suited to applications which require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. The WWDG is best suited to applications which require the watchdog to react within an accurate timing window. For further information on the window watchdog, refer to *Section 22 on page 713*.

21.2 IWDG main features

- Free-running downcounter
- clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Reset (if watchdog activated) when the downcounter value of 0x000 is reached

21.3 IWDG functional description

Figure 213 shows the functional blocks of the independent watchdog module.

When the independent watchdog is started by writing the value 0xCCCC in the Key register (IWDG_KR), the counter starts counting down from the reset value of 0xFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0xAAAA is written in the IWDG_KR register, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

21.3.1 Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and will generate a reset unless the Key register is written by the software before the counter reaches end of count.

21.3.2 Register access protection

Write access to the IWDG_PR and IWDG_RLR registers is protected. To modify them, first write the code 0x5555 in the IWDG_KR register. A write access to this register with a

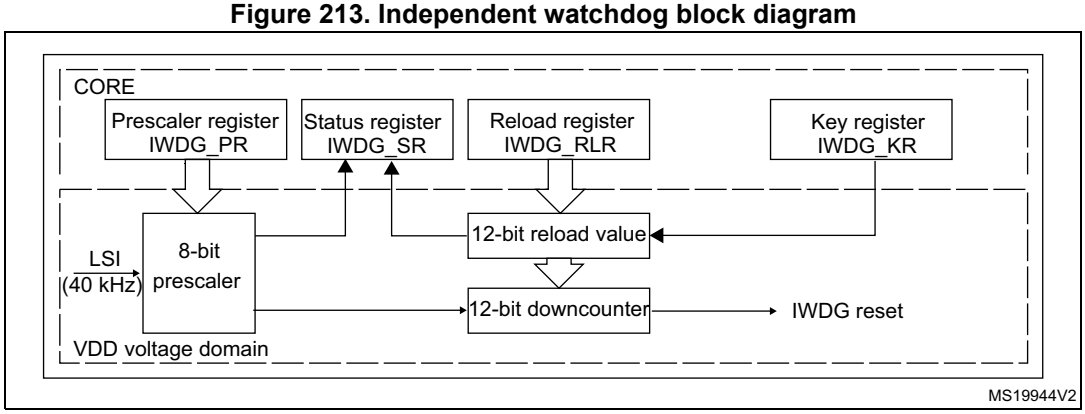
different value will break the sequence and register access will be protected again. This implies that it is the case of the reload operation (writing 0xAAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value is on going.

21.3.3 Debug mode

When the microcontroller enters debug mode (Cortex®-M4 with FPU core halted), the IWDG counter either continues to work normally or stops, depending on DBG_IWDG_STOP configuration bit in DBG module. For more details, refer to *Section 38.16.2: Debug support for timers, watchdog, bxCAN and I^2C*.

Figure 213. Independent watchdog block diagram



Note: The watchdog function is implemented in the V_{DD} voltage domain, still functional in Stop and Standby modes.

Table 107. Min/max IWDG timeout period (in ms) at 32 kHz (LSI)(1)

Prescaler divider	PR[2:0] bits	Min timeout RL[11:0]= 0x000	Max timeout RL[11:0]= 0xFFF
/4	0	0.125	512
/8	1	0.25	1024
/16	2	0.5	2048
/32	3	1	4096
/64	4	2	8192
/128	5	4	16384
/256	6	8	32768

1. These timings are given for a 32 kHz clock but the microcontroller internal RC frequency can vary. Refer to the LSI oscillator characteristics table in the device datasheet for maximum and minimum values.

21.4 IWDG registers

Refer to *Section 1.1* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by half-words (16 bits) or words (32 bits).

21.4.1 Key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000 (reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																KEY[15:0]															
																w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0000h)

These bits must be written by software at regular intervals with the key value AAAAh, otherwise the watchdog generates a reset when the counter reaches 0.
Writing the key value 5555h to enable access to the IWDG_PR and IWDG_RLR registers (see *Section 21.3.2*)
Writing the key value CCCCh starts the watchdog (except if the hardware watchdog option is selected)

21.4.2 Prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													PR[2:0]		
																													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **PR[2:0]:** Prescaler divider

These bits are write access protected see*Section 21.3.2*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of IWDG_SR must be reset in order to be able to change the prescaler divider.
000: divider /4
001: divider /8
010: divider /16
011: divider /32
100: divider /64
101: divider /128
110: divider /256
111: divider /256

Note: Reading this register returns the prescaler value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG_SR register is reset.

21.4.3 Reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF (reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				RL[11:0]											
																				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected see *Section 21.3.2*. They are written by software to define the value to be loaded in the watchdog counter each time the value AAAAh is written in the IWDG_KR register. The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Refer to *Table 107.*

The RVU bit in the IWDG_SR register must be reset in order to be able to change the reload value.

Note: Reading this register returns the reload value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on this register. For this reason the value read from this register is valid only when the RVU bit in the IWDG_SR register is reset.

21.4.4 Status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (not reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														RVU	PVU
																														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Reload value can be updated only when RVU bit is reset.

Bit 0 **PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Prescaler value can be updated only when PVU bit is reset.

Note: If several reload values or prescaler values are used by application, it is mandatory to wait until RVU bit is reset before changing the reload value and to wait until PVU bit is reset before changing the prescaler value. However, after updating the prescaler and/or the reload value it is not necessary to wait until RVU or PVU is reset before continuing code execution (even in case of low-power mode entry, the write operation is taken into account and will complete)

21.4.5 IWDG register map

The following table gives the IWDG register map and reset values.

Table 108. IWDG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**IWDG_KR**	Reserved																KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**IWDG_PR**	Reserved																													PR[2:0]		
	Reset value																														0	0	0
0x08	**IWDG_RLR**	Reserved																RL[11:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0C	**IWDG_SR**	Reserved																														RVU	PVU
	Reset value																															0	0

Refer to *Section 2.3: Memory map* for the register boundary addresses.

22 Window watchdog (WWDG)

This section applies to the whole STM32F4xx family, unless otherwise specified.

22.1 WWDG introduction

The window watchdog is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the downcounter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter has reached the window register value. This implies that the counter must be refreshed in a limited window.

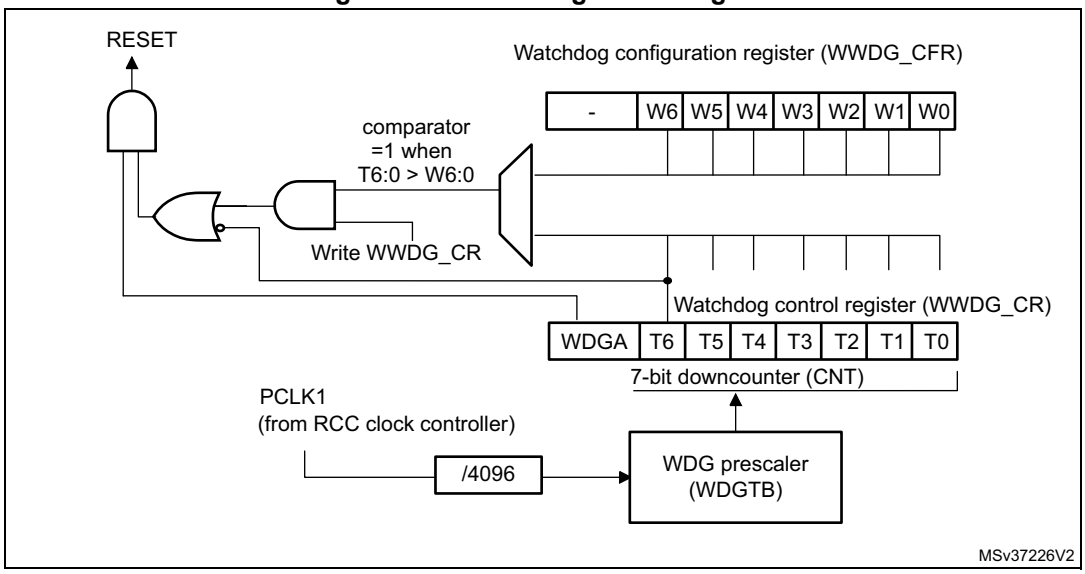
22.2 WWDG main features

- Programmable free-running downcounter
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes less than 0x40
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window (see *Figure 215*)
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the downcounter is equal to 0x40.

22.3 WWDG functional description

If the watchdog is activated (the WDGA bit is set in the WWDG_CR register) and when the 7-bit downcounter (T[6:0] bits) rolls over from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

Figure 214. Watchdog block diagram



The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the counter value is lower than the window register value. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Enabling the watchdog

The watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset.

Controlling the downcounter

This downcounter is free-running, 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 215*). 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 215* describes the window watchdog process.

Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

Advanced watchdog interrupt feature

The Early Wakeup Interrupt (EWI) can be used if specific safety operations or data logging must be performed before the actual reset is generated. The EWI interrupt is enabled by setting the EWI bit in the WWDG_CFR register. When the downcounter reaches the value 0x40, an EWI interrupt is generated and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case, the corresponding interrupt service routine (ISR) should reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

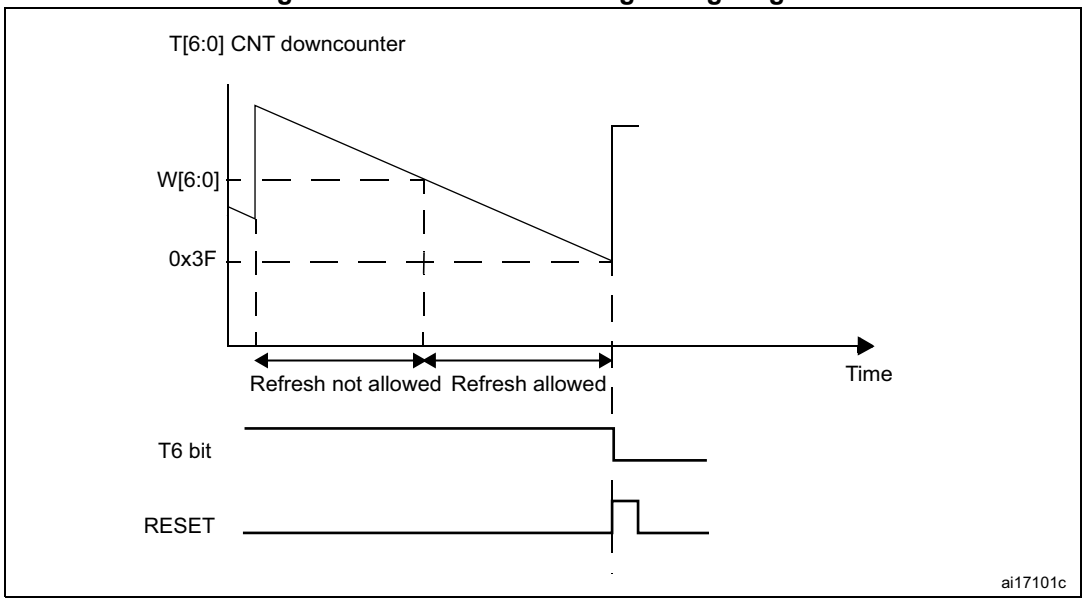
The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDG_SR register.

Note: When the EWI interrupt cannot be served, e.g. due to a system lock in a higher priority task, the WWDG reset will eventually be generated.

22.4 How to program the watchdog timeout

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 215. Window watchdog timing diagram



The formula to calculate the WWDG timeout value is given by:

$$t_{WWDG} = t_{PCLK1} \times 4096 \times 2^{WDGTB[1:0]} \times (T[5:0] + 1) \quad (ms)$$

where:

t_{WWDG}: WWDG timeout

t_{PCLK1}: APB1 clock period measured in ms

4096: value corresponding to internal divider

As an example, let us assume APB1 frequency is equal to 24 MHz, WDGTB[1:0] is set to 3 and T[5:0] is set to 63:

$$t_{WWDG} = 1/\ 24000 \times 4096 \times 2^{3} \times (63+1) = 21.85ms$$

Refer to *Table 109* for the minimum and maximum values of the t_{WWDG}.

Table 109. Minimum and maximum timeout values at 30 MHz (f_{PCLK1})

Prescaler	WDGTB	Min timeout (µs) T[5:0] = 0x00	Max timeout (ms) T[5:0] = 0x3F
1	0	136.53	8.74
2	1	273.07	17.48
4	2	546.13	34.95
8	3	1092.27	69.91

22.5 Debug mode

When the microcontroller enters debug mode (Cortex®-M4 with FPU core halted), the WWDG counter either continues to work normally or stops, depending on DBG_WWDG_STOP configuration bit in DBG module. For more details, refer to *Section 38.16.2: Debug support for timers, watchdog, bxCAN and I^2C*.

22.6 WWDG registers

Refer to *Section 1.1* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by half-words (16 bits) or words (32 bits).

22.6.1 Control register (WWDG_CR)

Address offset: 0x00

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								WDGA	T[6:0]						
								rs	rw						

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter. It is decremented every (4096 x $2^{WDGTB[1:0]}$) PCLK1 cycles. A reset is produced when it rolls over from 0x40 to 0x3F (T6 becomes cleared).

22.6.2 Configuration register (WWDG_CFR)

Address offset: 0x04

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						EWI	WDGTB[1:0]		W[6:0]						
						rs	rw		rw						

Bit 31:10 Reserved, must be kept at reset value.

Bit 9 **EWI:** Early wakeup interrupt

When set, an interrupt occurs whenever the counter reaches the value 0x40. This interrupt is only cleared by hardware after a reset.

Bits 8:7 **WDGTB[1:0]:** Timer base

The time base of the prescaler can be modified as follows:
00: CK Counter Clock (PCLK1 div 4096) div 1
01: CK Counter Clock (PCLK1 div 4096) div 2
10: CK Counter Clock (PCLK1 div 4096) div 4
11: CK Counter Clock (PCLK1 div 4096) div 8

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared to the downcounter.

22.6.3 Status register (WWDG_SR)

Address offset: 0x08
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															EWIF
															rc_w0

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’. A write of ‘1’ has no effect. This bit is also set if the interrupt is not enabled.

22.6.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 110. WWDG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**WWDG_CR**	Reserved																								WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x04	**WWDG_CFR**	Reserved																						EWI	WDGTB1	WDGTB0	W[6:0]						
	Reset value																							0	0	0	1	1	1	1	1	1	1
0x08	**WWDG_SR**	Reserved																															EWIF
	Reset value																																0

Refer to *Section 2.3: Memory map* for the register boundary addresses.

23 Cryptographic processor (CRYP)

This section applies to STM32F415/417xx and STM32F43xxx devices.

23.1 CRYP introduction

The cryptographic processor can be used to both encipher and decipher data using the DES, Triple-DES or AES (128, 192, or 256) algorithms. It is a fully compliant implementation of the following standards:

- The data encryption standard (DES) and Triple-DES (TDES) as defined by Federal Information Processing Standards Publication (FIPS PUB 46-3, 1999 October 25). It follows the American National Standards Institute (ANSI) X9.52 standard.
- The advanced encryption standard (AES) as defined by Federal Information Processing Standards Publication (FIPS PUB 197, 2001 November 26)

The CRYP processor performs data encryption and decryption using DES and TDES algorithms in Electronic codebook (ECB) or Cipher block chaining (CBC) mode.

The CRYP peripheral is a 32-bit AHB2 peripheral. It supports DMA transfer for incoming and processed data, and has input and output FIFOs (each 8 words deep).

23.2 CRYP main features

- Suitable for AES, DES and TDES enciphering and deciphering operations
- AES
  - Supports the ECB, CBC, CTR, CCM and GCM chaining algorithms (CCM and GCM are available on STM32F42xxx and STM32F43xxx only)
  - Supports 128-, 192- and 256-bit keys
  - 4 × 32-bit initialization vectors (IV) used in the CBC, CTR, CCM and GCM modes

Table 111. Number of cycles required to process each 128-bit block (STM32F415/417xx)

Algorithm / Key size	ECB	CBC	CTR
128b	14	14	14
192b	16	16	16
256b	18	18	18

Table 112. Number of cycles required to process each 128-bit block (STM32F43xxx)

Algorithm / Key size	ECB	CBC	CTR	GCM				CCM			
				Init	Header	Payload	Tag	Init	Header	Payload	Tag
128b	14	14	14	24	10	14	14	12	14	25	14

Table 112. Number of cycles required to process each 128-bit block (STM32F43xxx)

192b	16	16	16	28	10	16	16	14	16	29	16
256b	18	18	18	32	10	18	18	16	18	33	18

- DES/TDES
  - Direct implementation of simple DES algorithms (a single key, K1, is used)
  - Supports the ECB and CBC chaining algorithms
  - Supports 64-, 128- and 192-bit keys (including parity)
  - 2 × 32-bit initialization vectors (IV) used in the CBC mode
  - 16 HCLK cycles to process one 64-bit block in DES
  - 48 HCLK cycles to process one 64-bit block in TDES
- Common to DES/TDES and AES
  - IN and OUT FIFO (each with an 8-word depth, a 32-bit width, corresponding to 4 DES blocks or 2 AES blocks)
  - Automatic data flow control with support of direct memory access (DMA) (using 2 channels, one for incoming data the other for processed data)
  - Data swapping logic to support 1-, 8-, 16- or 32-bit data

23.3 CRYP functional description

The cryptographic processor implements a Triple-DES (TDES, that also supports DES) core and an AES cryptographic core. *Section 23.3.1* and *Section 23.3.2* provide details on these cores.

Since the TDES and the AES algorithms use block ciphers, incomplete input data blocks have to be padded prior to encryption (extra bits should be appended to the trailing end of the data string). After decryption, the padding has to be discarded. The hardware does not manage the padding operation, the software has to handle it.

Figure 216 shows the block diagram of the cryptographic processor.

Figure 216. Block diagram (STM32F415/417xx)

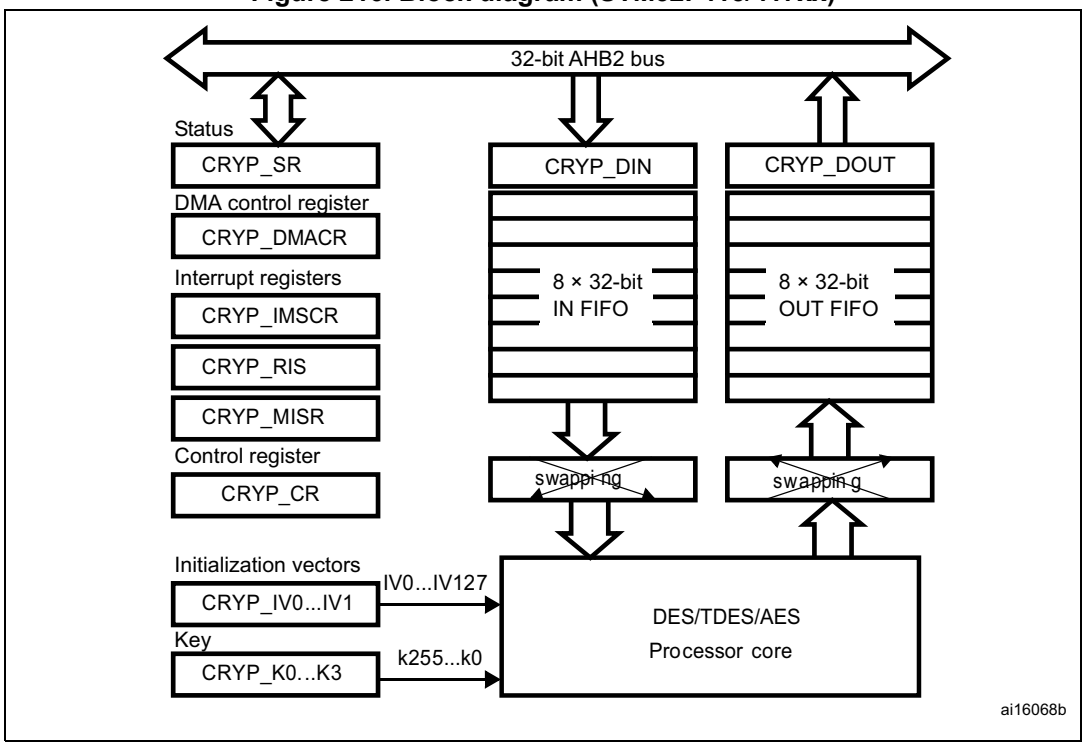
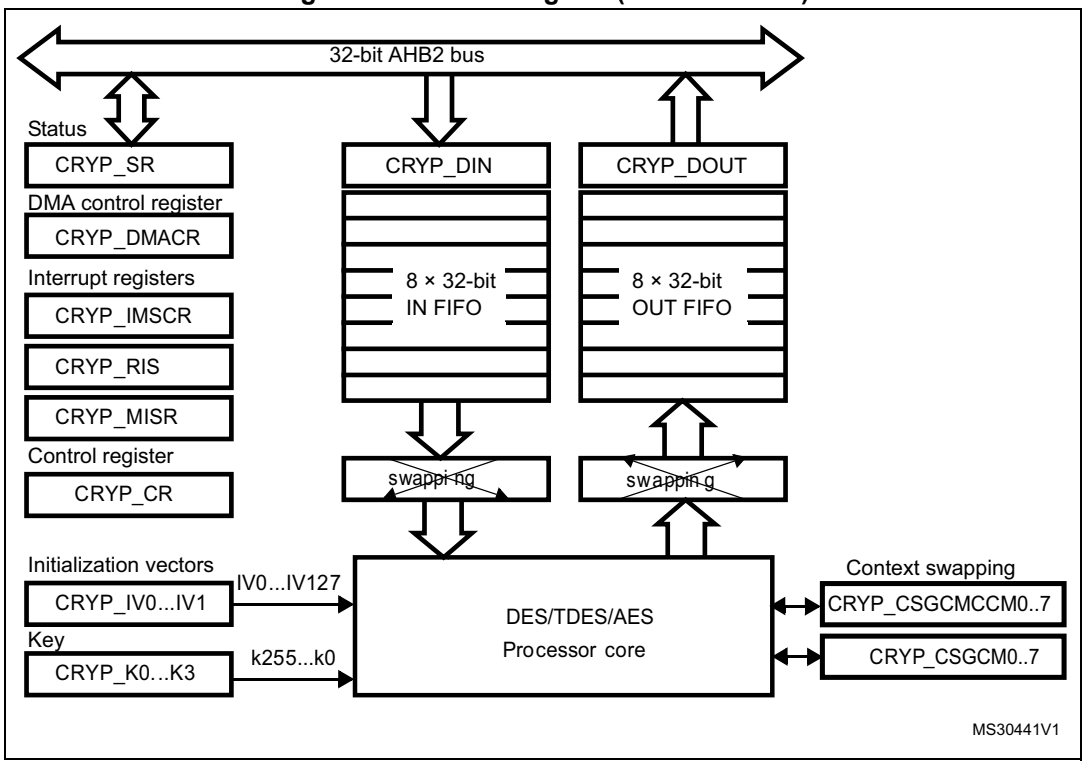


Figure 217. Block diagram (STM32F43xxx)



23.3.1 DES/TDES cryptographic core

The DES/Triple-DES cryptographic core consists of three components:

- The DES algorithm (DEA)
- Multiple keys (1 for the DES algorithm, 1 to 3 for the TDES algorithm)
- The initialization vector (used in the CBC mode)

The basic processing involved in the TDES is as follows: an input block is read in the DEA and encrypted using the first key, K1 (K0 is not used in TDES mode). The output is then decrypted using the second key, K2, and encrypted using the third key, K3. The key depends on the algorithm which is used:

- DES mode: Key = [K1]
- TDES mode: Key = [K3 K2 K1]

where Kx=[KxR KxL], R = right, L = left

According to the mode implemented, the resultant output block is used to calculate the ciphertext.

Note that the outputs of the intermediate DEA stages is never revealed outside the cryptographic boundary.

The TDES allows three different keying options:

- Three independent keys

  The first option specifies that all the keys are independent, that is, K1, K2 and K3 are independent. FIPS PUB 46-3 – 1999 (and ANSI X9.52 – 1998) refers to this option as the Keying Option 1 and, to the TDES as 3-key TDES.

- Two independent keys

  The second option specifies that K1 and K2 are independent and K3 is equal to K1, that is, K1 and K2 are independent, K3 = K1. FIPS PUB 46-3 – 1999 (and ANSI X9.52 – 1998) refers to this second option as the Keying Option 2 and, to the TDES as 2-key TDES.

- Three equal keys

  The third option specifies that K1, K2 and K3 are equal, that is, K1 = K2 = K3. FIPS PUB 46-3 – 1999 (and ANSI X9.52 – 1998) refers to the third option as the Keying Option 3. This “1-key” TDES is equivalent to single DES.

FIPS PUB 46-3 – 1999 (and ANSI X9.52-1998) provides a thorough explanation of the processing involved in the four operation modes supplied by the TDEA (TDES algorithm): TDES-ECB encryption, TDES-ECB decryption, TDES-CBC encryption and TDES-CBC decryption.
This reference manual only gives a brief explanation of each mode.

DES and TDES Electronic codebook (DES/TDES-ECB) mode

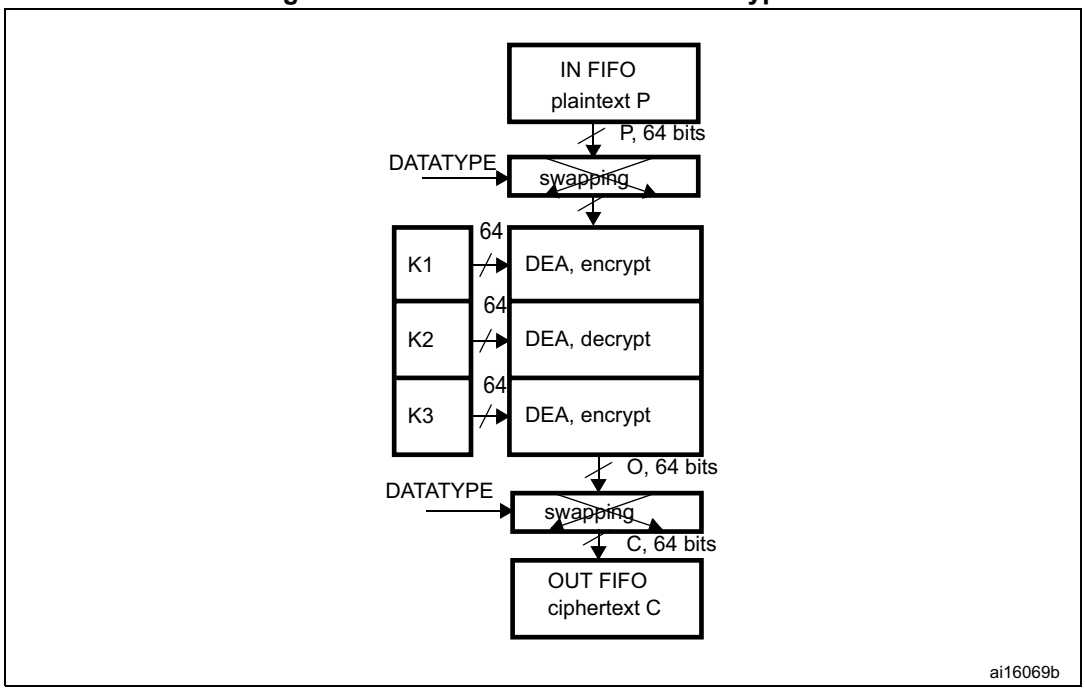
- DES/TDES-ECB mode encryption

  Figure 218 illustrates the encryption in DES and TDES Electronic codebook (DES/TDES-ECB) mode. A 64-bit plaintext data block (P) is used after bit/byte/half-word swapping (refer to *Section 23.3.3: Data type on page 739*) as the input block (I). The input block is processed through the DEA in the encrypt state using K1. The output of this process is fed back directly to the input of the DEA where the DES is performed in the decrypt state using K2. The output of this process is fed back directly to the input of the DEA where the DES is performed in the encrypt state using K3. The resultant 64-bit output block (O) is used, after bit/byte/half-word swapping, as ciphertext (C) and it is pushed into the OUT FIFO.

- DES/TDES-ECB mode decryption

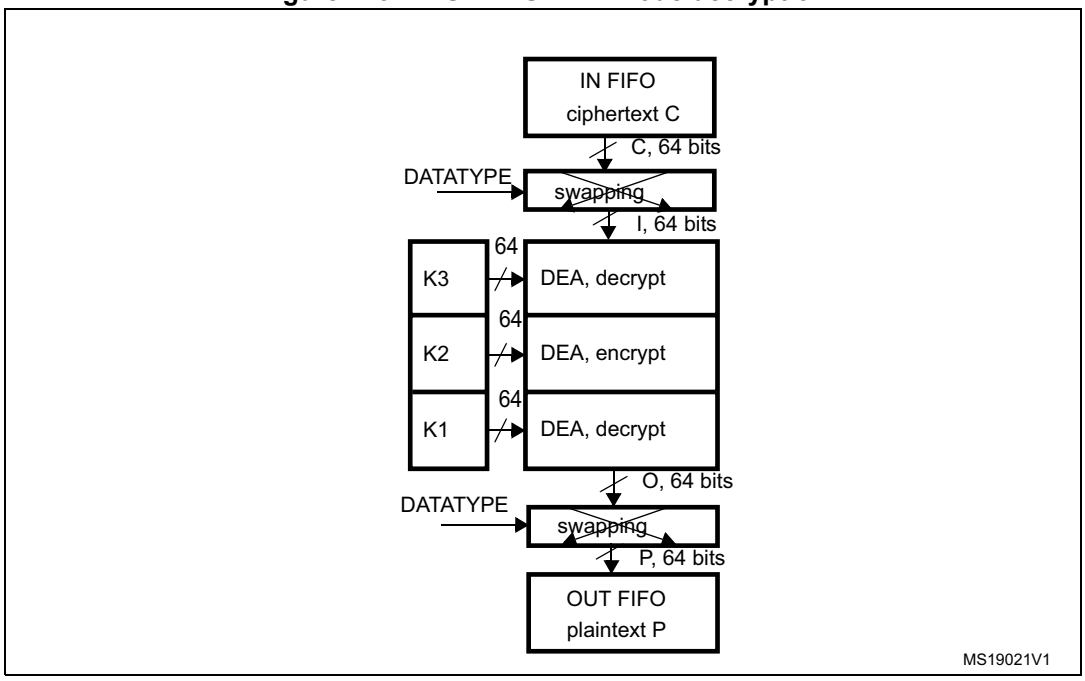
  Figure 219 illustrates the DES/TDES-ECB decryption. A 64-bit ciphertext block (C) is used, after bit/byte/half-word swapping, as the input block (I). The keying sequence is reversed compared to that used in the encryption process. The input block is processed through the DEA in the decrypt state using K3. The output of this process is fed back directly to the input of the DEA where the DES is performed in the encrypt state using K2. The new result is directly fed to the input of the DEA where the DES is performed in the decrypt state using K1. The resultant 64-bit output block (O), after bit/byte/half-word swapping, produces the plaintext (P).

Figure 218. DES/TDES-ECB mode encryption



1. K: key; C: cipher text; I: input block; O: output block; P: plain text.

Figure 219. DES/TDES-ECB mode decryption



1. K: key; C: cipher text; I: input block; O: output block; P: plain text.

DES and TDES Cipher block chaining (DES/TDES-CBC) mode

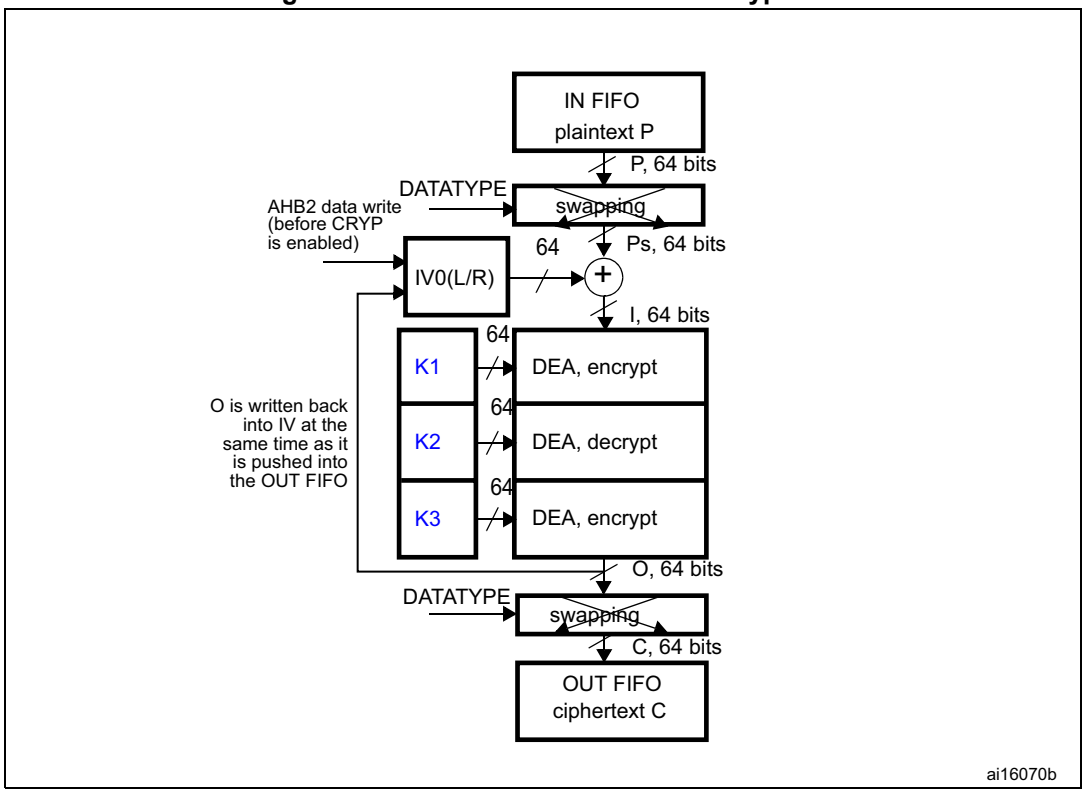
- DES/TDES-CBC mode encryption

  Figure 220 illustrates the DES and Triple-DES Cipher block chaining (DES/TDES-CBC) mode encryption. This mode begins by dividing a plaintext message into 64-bit data blocks. In TCBC encryption, the first input block (I_1), obtained after bit/byte/half-word swapping (refer to *Section 23.3.3: Data type on page 739*), is formed by exclusive-ORing the first plaintext data block (P_1) with a 64-bit initialization vector IV (I_1 = IV $\oplus$ P_1). The input block is processed through the DEA in the encrypt state using K1. The output of this process is fed back directly to the input of the DEA, which performs the DES in the decrypt state using K2. The output of this process is fed directly to the input of the DEA, which performs the DES in the encrypt state using K3. The resultant 64-bit output block (O_1) is used directly as the ciphertext (C_1), that is, $C_1 = O_1$. This first ciphertext block is then exclusive-ORed with the second plaintext data block to produce the second input block, (I_2) = ($C_1 \oplus P_2$). Note that I_2 and P_2 now refer to the second block. The second input block is processed through the TDEA to produce the second ciphertext block. This encryption process continues to “chain” successive cipher and plaintext blocks together until the last plaintext block in the message is encrypted. If the message does not consist of an integral number of data blocks, then the final partial data block should be encrypted in a manner specified for the application.

- DES/TDES-CBC mode decryption

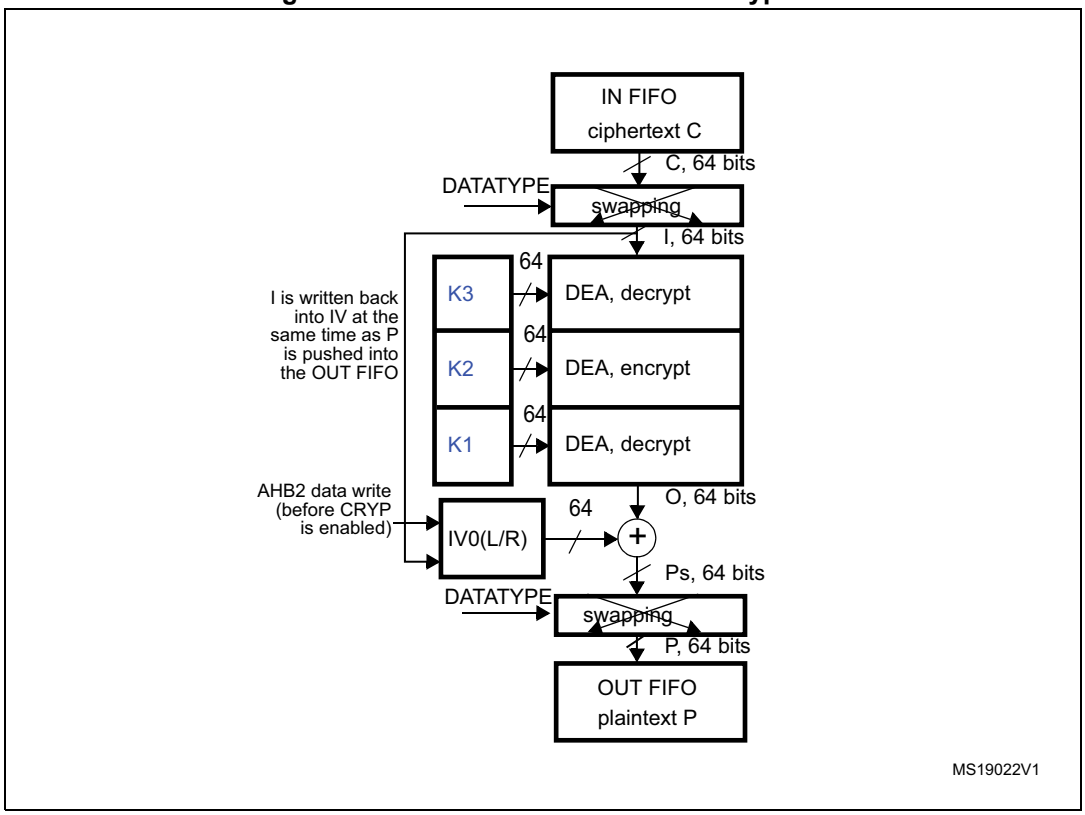
  In DES/TDES-CBC decryption (see *Figure 221*), the first ciphertext block (C_1) is used directly as the input block (I_1). The keying sequence is reversed compared to that used for the encrypt process. The input block is processed through the DEA in the decrypt state using K3. The output of this process is fed directly to the input of the DEA where the DES is processed in the encrypt state using K2. This resulting value is directly fed to the input of the DEA where the DES is processed in the decrypt state using K1. The resulting output block is exclusive-ORed with the IV (which must be the same as that used during encryption) to produce the first plaintext block ($P_1 = O_1 \oplus$ IV). The second ciphertext block is then used as the next input block and is processed through the TDEA. The resulting output block is exclusive-ORed with the first ciphertext block to produce the second plaintext data block ($P_2 = O_2 \oplus C_1$). (Note that P_2 and O_2 refer to the second block of data.) The TCBC decryption process continues in this manner until the last complete ciphertext block has been decrypted. Ciphertext representing a partial data block must be decrypted in a manner specified for the application.

Figure 220. DES/TDES-CBC mode encryption



1. K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: initialization vectors.

Figure 221. DES/TDES-CBC mode decryption



1. K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: initialization vectors.

23.3.2 AES cryptographic core

The AES cryptographic core consists of three components:

- The AES algorithm (AEA: advanced encryption algorithm)
- Multiple keys
- Initialization vector(s) or Nonce

The AES utilizes keys of 3 possible lengths: 128, 192 or 256 bits and, depending on the operation mode used, zero or one 128-bit initialization vector (IV).

The basic processing involved in the AES is as follows: an input block of 128 bits is read from the input FIFO and sent to the AEA to be encrypted using the key (K0...3). The key format depends on the key size:

- If Key size = 128: Key = [K3 K2]
- If Key size = 192: Key = [K3 K2 K1]
- If Key size = 256: Key = [K3 K2 K1 K0]

where Kx=[KxR KxL],R=right, L=left

According to the mode implemented, the resultant output block is used to calculate the ciphertext.

FIPS PUB 197 (November 26, 2001) provides a thorough explanation of the processing involved in the four operation modes supplied by the AES core: AES-ECB encryption, AES-

ECB decryption, AES-CBC encryption and AES-CBC decryption.This reference manual only gives a brief explanation of each mode.

AES Electronic codebook (AES-ECB) mode

- AES-ECB mode encryption

  Figure 222 illustrates the AES Electronic codebook (AES-ECB) mode encryption.

  In AES-ECB encryption, a 128- bit plaintext data block (P) is used after bit/byte/half-word swapping (refer to *Section 23.3.3: Data type on page 739*) as the input block (I). The input block is processed through the AEA in the encrypt state using the 128, 192 or 256-bit key. The resultant 128-bit output block (O) is used after bit/byte/half-word swapping as ciphertext (C). It is then pushed into the OUT FIFO.

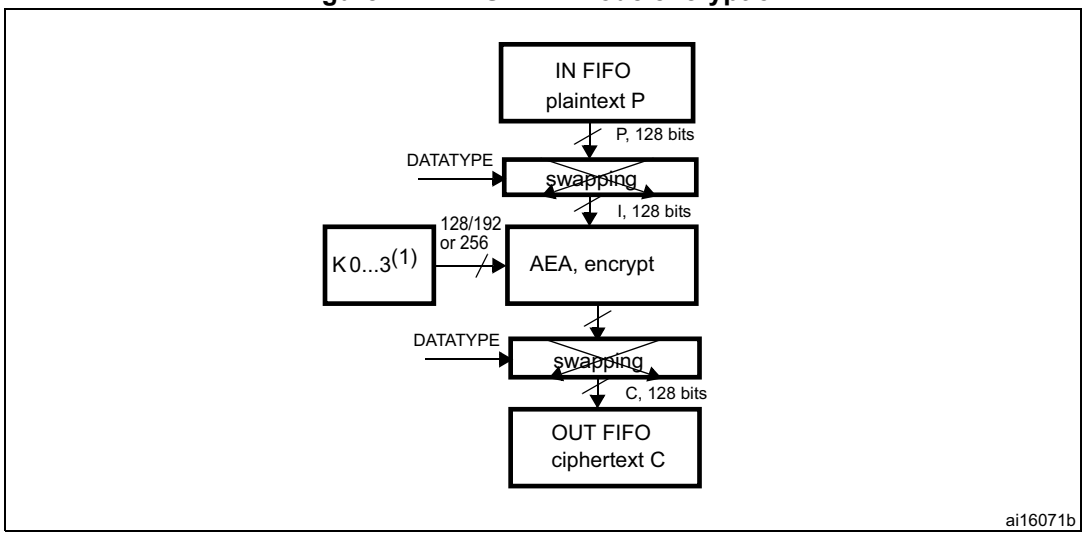
- AES-ECB mode decryption

  Figure 223 illustrates the AES Electronic codebook (AES-ECB) mode encryption.

  To perform an AES decryption in the ECB mode, the secret key has to be prepared (it is necessary to execute the complete key schedule for encryption) by collecting the last round key, and using it as the first round key for the decryption of the ciphertext. This preparation function is computed by the AES core. Refer to *Section 23.3.6: Procedure to perform an encryption or a decryption* for more details on how to prepare the key.

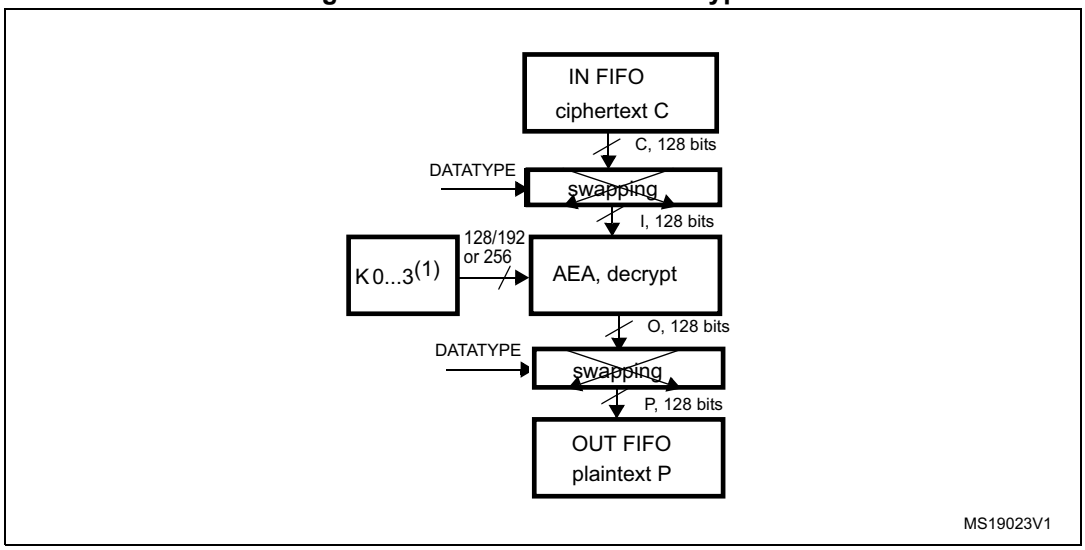
  In AES-ECB decryption, a 128-bit ciphertext block (C) is used after bit/byte/half-word swapping as the input block (I). The keying sequence is reversed compared to that of the encryption process. The resultant 128-bit output block (O), after bit/byte or half-word swapping, produces the plaintext (P).

Figure 222. AES-ECB mode encryption



1. K: key; C: cipher text; I: input block; O: output block; P: plain text.
2. If Key size = 128: Key = [K3 K2].
   If Key size = 192: Key = [K3 K2 K1]
   If Key size = 256: Key = [K3 K2 K1 K0].

Figure 223. AES-ECB mode decryption



1. K: key; C: cipher text; I: input block; O: output block; P: plain text.
2. If Key size = 128 => Key = [K3 K2].
   If Key size = 192 => Key = [K3 K2 K1]
   If Key size = 256 => Key = [K3 K2 K1 K0].

AES Cipher block chaining (AES-CBC) mode

- AES-CBC mode encryption

  The AES Cipher block chaining (AES-CBC) mode decryption is shown on *Figure 224*.

  In AES-CBC encryption, the first input block (I_1) obtained after bit/byte/half-word swapping (refer to *Section 23.3.3: Data type on page 739*) is formed by exclusive-ORing the first plaintext data block (P_1) with a 128-bit initialization vector IV ($I_1 = IV \oplus P_1$). The input block is processed through the AEA in the encrypt state using the 128-, 192- or 256-bit key (K0...K3). The resultant 128-bit output block (O_1) is used directly as ciphertext (C_1), that is, $C_1 = O_1$. This first ciphertext block is then exclusive-ORed with the second plaintext data block to produce the second input block, (I_2) = ($C_1 \oplus P_2$). Note that I_2 and P_2 now refer to the second block. The second input block is processed through the AEA to produce the second ciphertext block. This encryption process continues to “chain” successive cipher and plaintext blocks together until the last plaintext block in the message is encrypted. If the message does not consist of an integral number of data blocks, then the final partial data block should be encrypted in a manner specified for the application.

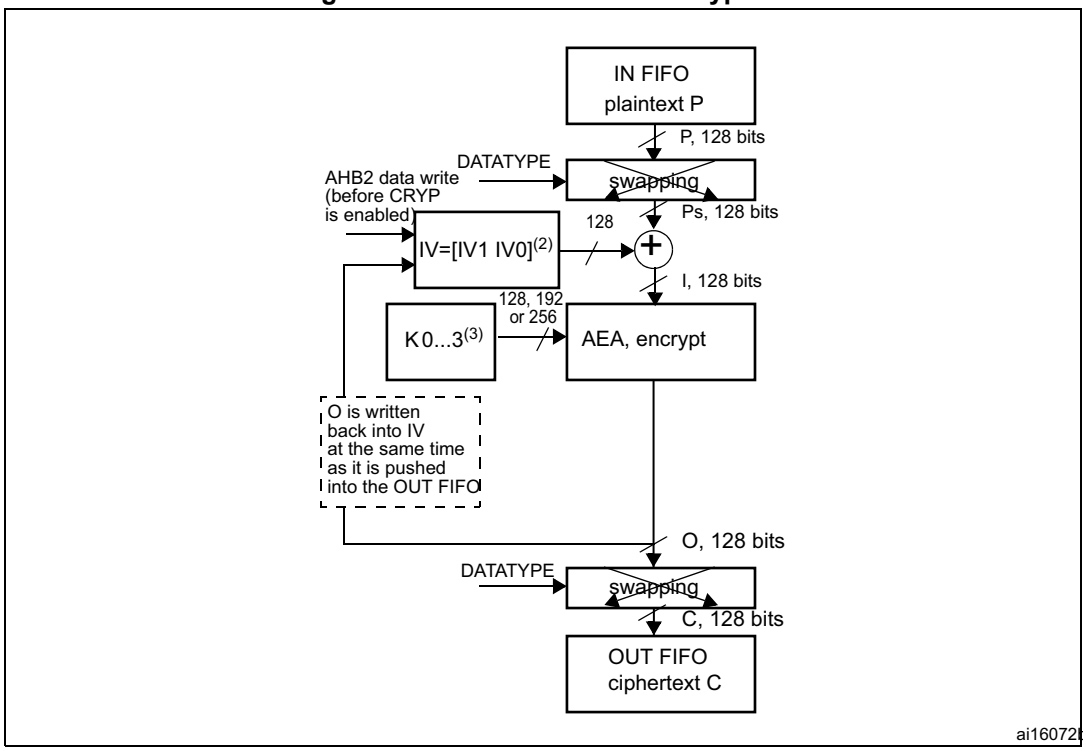
  In the CBC mode, like in the ECB mode, the secret key must be prepared to perform an AES decryption. Refer to *Section 23.3.6: Procedure to perform an encryption or a decryption on page 744* for more details on how to prepare the key.

- AES-CBC mode decryption

  In AES-CBC decryption (see *Figure 225*), the first 128-bit ciphertext block (C_1) is used directly as the input block (I_1). The input block is processed through the AEA in the decrypt state using the 128-, 192- or 256-bit key. The resulting output block is exclusive-ORed with the 128-bit initialization vector IV (which must be the same as that used during encryption) to produce the first plaintext block ($P_1 = O_1 \oplus IV$). The second ciphertext block is then used as the next input block and is processed through the AEA. The resulting output block is exclusive-ORed with the first ciphertext block to produce the second plaintext data block ($P_2 = O_2 \oplus C_1$). (Note that P_2 and O_2 refer to the second

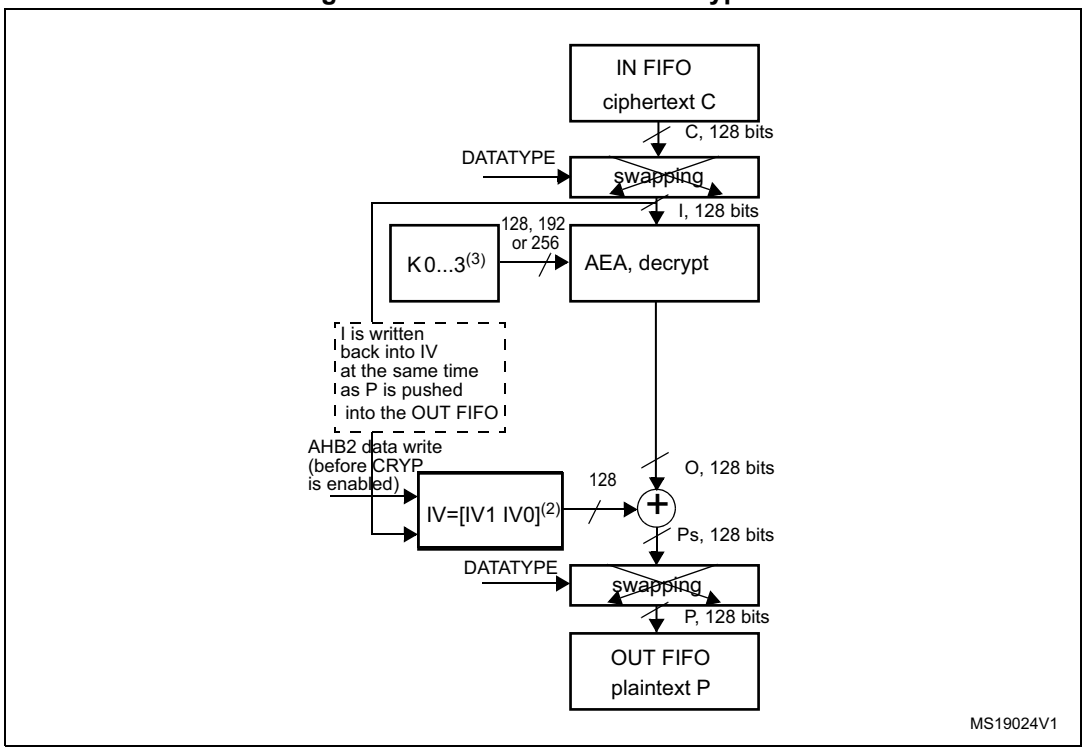
block of data.) The AES-CBC decryption process continues in this manner until the last complete ciphertext block has been decrypted. Ciphertext representing a partial data block must be decrypted in a manner specified for the application.

Figure 224. AES-CBC mode encryption



1. K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: Initialization vectors.
2. IVx=[IVxR IVxL], R=right, L=left.
3. If Key size = 128 => Key = [K3 K2].
   If Key size = 192 => Key = [K3 K2 K1]
   If Key size = 256 => Key = [K3 K2 K1 K0].

Figure 225. AES-CBC mode decryption



1. K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: Initialization vectors.
2. IVx=[IVxR IVxL], R=right, L=left.
3. If Key size = 128 => Key = [K3 K2].
   If Key size = 192 => Key = [K3 K2 K1]
   If Key size = 256 => Key = [K3 K2 K1 K0].

AES counter mode (AES-CTR) mode

The AES counter mode uses the AES block as a key stream generator. The generated keys are then XORed with the plaintext to obtain the cipher. For this reason, it makes no sense to speak of different CTR encryption/decryption, since the two operations are exactly the same.

In fact, given:

- Plaintext: P[0], P[1], ..., P[n] (128 bits each)
- A key K to be used (the size does not matter)
- An initial counter block (call it ICB but it has the same functionality as the IV of CBC)

The cipher is computed as follows:

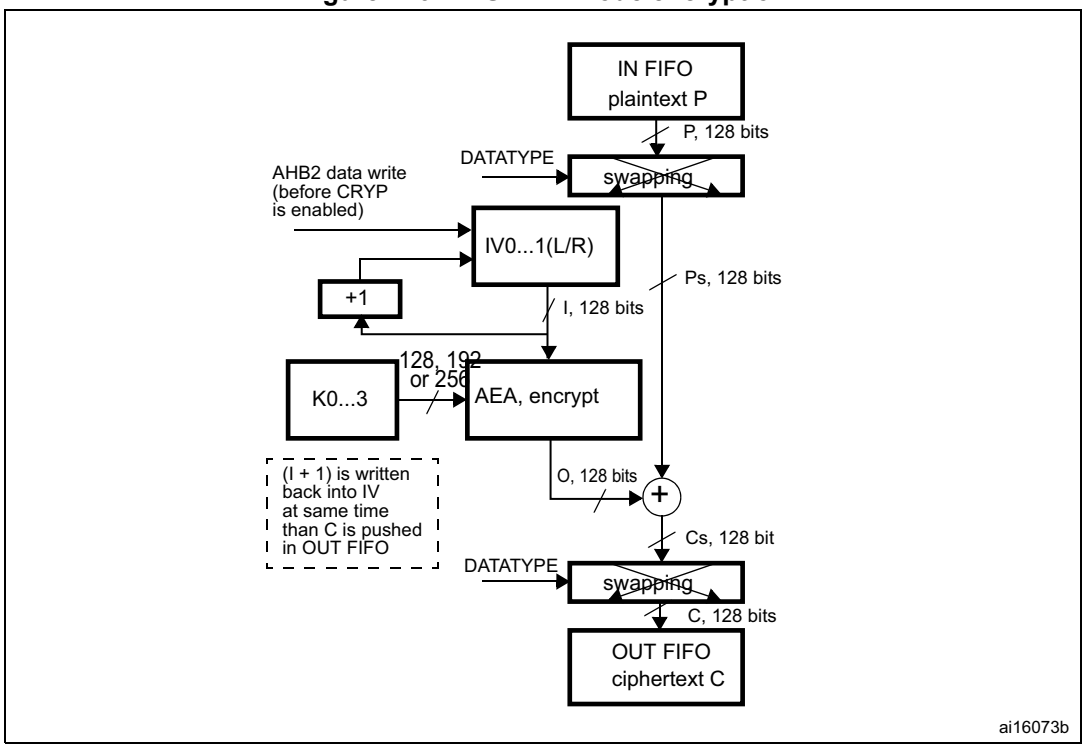
`C[i] = enck(iv[i]) xor P[i]`, where:

`iv[0]` = ICB and `iv[i+1]` = `func(iv[i])`, where `func` is an update function applied to the previous iv block; `func` is basically an increment of one of the fields composing the iv block.

Given that the ICB for decryption is the same as the one for encryption, the key stream generated during decryption is the same as the one generated during encryption. Then, the ciphertext is XORed with the key stream in order to retrieve the original plaintext. The decryption operation therefore acts exactly in the same way as the encryption operation.

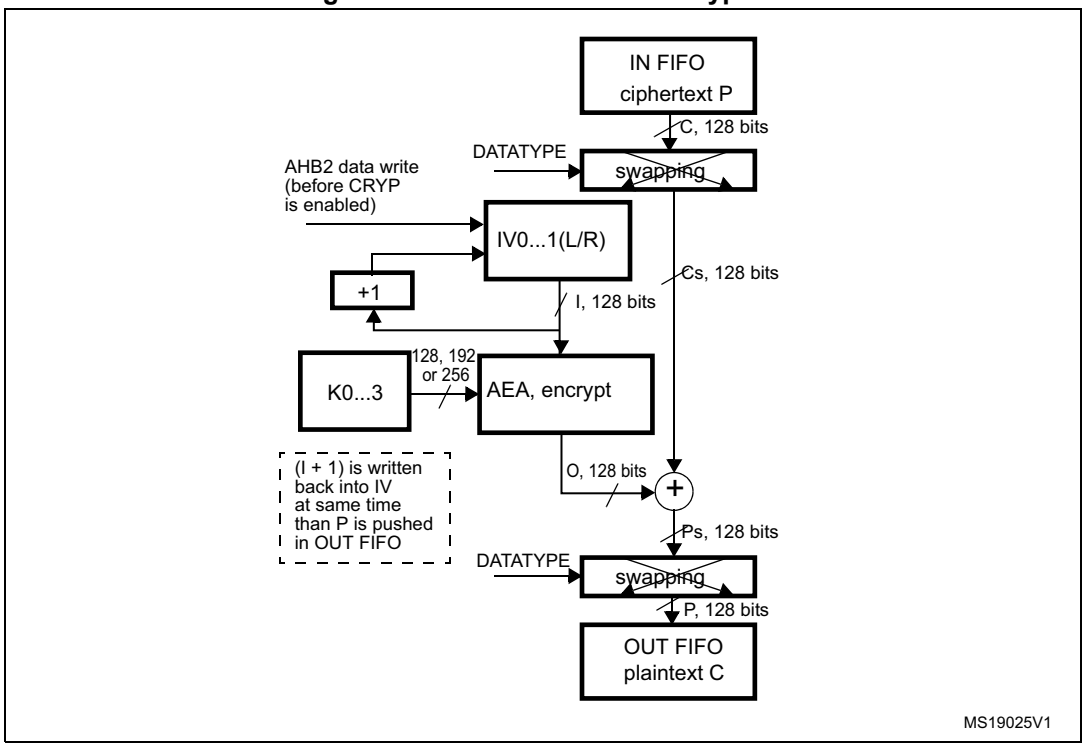
Figure 226 and *Figure 227* illustrate AES-CTR encryption and decryption, respectively.

Figure 226. AES-CTR mode encryption



1. K: key; C: cipher text; I: input Block; o: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); Cs: cipher text after swapping (when decoding) or before swapping (when encoding); P: plain text; IV: Initialization vectors.

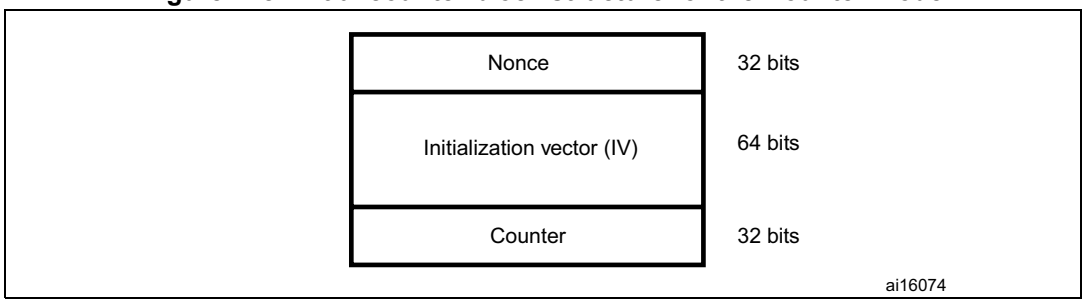
Figure 227. AES-CTR mode decryption



1. K: key; C: cipher text; I: input Block; o: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); Cs: cipher text after swapping (when decoding) or before swapping (when encoding); P: plain text; IV: Initialization vectors.

Figure 228 shows the structure of the IV block as defined by the standard [2]. It is composed of three distinct fields.

Figure 228. Initial counter block structure for the Counter mode



- Nonce is a 32-bit, single-use value. A new nonce should be assigned to each different communication.
- The initialization vector (IV) is a 64-bit value and the standard specifies that the encryptor must choose IV so as to ensure that a given value is used only once for a given key
- The counter is a 32-bit big-endian integer that is incremented each time a block has been encrypted. The initial value of the counter should be set to ‘1’.

The block increments the least significant 32 bits, while it leaves the other (most significant) 96 bits unchanged.

AES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) allows encrypting and authenticating the plaintext, and generating the correspondent ciphertext and tag (also known as message authentication code or message integrity check). This algorithm is based on AES counter mode to ensure confidentiality. It uses a multiplier over a fixed finite field to generate the tag. An initialization vector is required at the beginning of the algorithm.

The message to be processed is split into 2 parts:

- The header (also knows as additional authentication data): data which is authenticated but no protected (such as information for routing the packet)
- The payload (also knows as plaintext or ciphertext): the message itself which is authenticated and encrypted.

Note: The header must precede the payload and the two parts cannot be mixed together.

The GCM standard requires to pass, at the end of the message, a specific 128-bit block composed of the size of the header (64 bits) and the size of the payload (64 bits). During the computation, the header blocks must be distinguished from the payload blocks.

In GCM mode, four steps are required to perform an encryption/decryption:

1. GCM init phase

   During this first step, the HASH key is calculated and saved internally to be used for processing all the blocks. It is recommended to follow the sequence below:

   a) Make sure that the cryptographic processor is disabled by clearing the CRYPEN bit in the CRYP_CR register.

   b) Select the GCM chaining mode by programming ALGOMODE bits to ‘01000’ in CRYP_CR.

   c) Configure GCM_CCMPH bits to ‘00’ in CRYP_CR to start the GCM Init phase.

   d) Initialize the key registers (128,192 and 256 bits) in CRYP_KEYRx as well as the initialization vector (IV).

   e) Set CRYPEN bit to ‘1’ to start the calculation of the HASH key.

   f) Wait for the CRYPEN bit to be cleared to ‘0’ before moving on to the next phase.

   g) Set the CRYPEN bit to ‘1’.

2. GCM header phase

   This step must be performed after the GCM Init phase:

   h) Set the GCM_CCMPH bits to ‘01’ in CRYP_CR to indicate that the header phase has started.

   i) Write the header data. Three methods can be used:

   – Program the data by blocks of 32 bits into the CRYP_DIN register, and use the IFNF flag to determine if the input FIFO can receive data. The size of the header must be a multiple of 128 bits (4 words).

   – Program the data into the CRYP_DIN register by blocks of 8 words, and use the IFEM flag to determine if the input FIFO can receive data (IFEM=’1’). The size of the header must be a multiple of 128 bits (4 words).

   – Use the DMA.

   j) Once all header data have been supplied, wait until the BUSY bit is cleared in the CRYP_SR register.

3. GCM payload phase (encryption/decryption)

   This step must be performed after the GCM header phase:

   k) Configure GCM_CCMPH to ‘10’ in the CRYP_CR register.

   l) Select the algorithm direction (encryption or decryption) by using the ALGODIR bit in CRYP_CR.

   m) Program the payload message into the CRYP_DIN register, and use the IFNF flag to determine if the input FIFO can receive data. Alternatively, the data could be programmed into the CRYP_DIN register by blocks of 8 words and the IFEM flag used to determine if the input FIFO can receive data (IFEM=’1’). In parallel, the

OFNE/OFFU flag of the CRYP_DOUT register can be monitored to check if the output FIFO is not empty.

n) Repeat the previous step until all payload blocks have been encrypted or decrypted. Alternatively, DMA could be used.

4. GCM final phase

   This step generates the authentication tag:

   o) Configure GCM_CCMPH[1:0] to '11' in CRYP_CR.

   p) Write the input into the CRYP_DIN register 4 times. The input must contain the number of bits in the header (64 bits) concatenated with the number of bits in the payload (64 bits).

   q) Wait till the OFNE flag (FIFO output not empty) is set to '1' in the CRYP_SR register.

   r) Read the CRYP_DOUT register 4 times: the output corresponds to the authentication tag.

   s) Disable the cryptographic processor (CRYPEN bit in CRYP_CR = '0')

Note: When a decryption is performed, it is not required to compute the key at the beginning. At the end of the decryption, the generated tag should be compared with the expected tag passed with the message. In addition, the ALGODIR bit (algorithm direction) must be set to '1'.

No need to disable/enable CRYP processor when moving from header phase to tag phase.

AES Galois message authentication code (GMAC)

The cryptographic processor also supports GMAC to authenticate the plaintext. It uses the GCM algorithm and a multiplier over a fixed finite field to generate the corresponding tag.

An initialization vector is required at the beginning of the algorithm.

Actually, the GMAC algorithm corresponds to the GCM algorithm applied on a message composed of the header only. As a consequence, the payload phase is not required.

AES combined cipher machine (CCM)

The CCM algorithm allows encrypting and authenticating the plaintext, as well as generating the corresponding ciphertext and tag (also known as message authentication code or message integrity check). This algorithm is based on AES counter mode to ensure confidentiality. It uses the AES CBC mode to generate a 128-bit tag.

The CCM standard (RFC 3610 Counter with CBC-MAC (CCM) dated September 2003) defines particular encoding rules for the first authentication block (called B0 in the standard). In particular, the first block includes flags, a nonce and the payload length expressed in bytes. The CCM standard specifies another format, called A or counter, for encryption/decryption. The counter is incremented during the payload phase and its 32 LSB bits are initialized to '1' during the tag generation (called A0 packet in the CCM standard).

Note: The hardware does not perform the formatting operation of the B0 packet. It should be handled by the software.

As for the GCM algorithm, the message to be processed is split into 2 parts:

- The header (also knows as additional authentication data): data which is authenticated but no protected (such as information for routing the packet)
- The payload (also knows as plaintext or ciphertext): the message itself which is authenticated and encrypted.

Note: The header part must precede the payload and the two parts cannot be mixed together.

In CCM mode, 4 steps are required to perform and encryption or decryption:

1. CCM init phase

   In this first step, the B0 packet of the CCM message (1st packet) is programmed into the CRYP_DIN register. During this phase, the CRYP_DOUT register does not contain any output data.

   The following sequence must be followed:

   a) Make sure that the cryptographic processor is disabled by clearing the CRYPEN bit in the CRYP_CR register.

   b) Select the CCM chaining mode by programming the ALGOMODE bits to ‘01001’ in the CRYP_CR register.

   c) Configure the GCM_CCMPH bits to ‘00’ in CRYP_CR to start the CCM Init phase.

   d) Initialize the key registers (128,192 and 256 bits) in CRYP_KEYRx as well as the initialization vector (IV).

   e) Set the CRYPEN bit to ‘1’ in CRYP_CR.

   f) Program the B0 packet into the input data register.

   g) Wait for the CRYPEN bit to be cleared before moving on to the next phase.

   h) Set CRYPEN to ‘1’.

2. CCM header phase

   This step must be performed after the CCM Init phase. The sequence is identical for encryption and decryption.

   During this phase, the CRYP_DOUT register does not contain any output data.

   This phase can be skipped if there is no additional authenticated data.

   The following sequence must be followed:

   i) Set the GCM_CCMPH bit to ’01’ in CRYP_CR to indicate that the header phase has started.

   j) Three methods can be used:

   – Program the header data by blocks of 32 bits into the CRYP_DIN register, and use the IFNF flag to determine if the input FIFO can receive data. The size of the header must be a multiple of 128 bits (4 words).

   – Program the header data into the CRYP_DIN register by blocks of 8 words, and use the IFEM flag to determine if the input FIFO can receive data (IFEM=’1’). The size of the header must be a multiple of 128 bits (4 words).

   – Use the DMA.

Note: The first block B1 must be formatted with the header length. This task should be handled by software.

k) Once all header data have been supplied, wait until the BUSY flag is cleared.

3. CCM payload phase (encryption/decryption)

   This step must be performed after the CCM header phase. During this phase, the encrypted/decrypted payload is stored in the CRYP_DOUT register.

   The following sequence must be followed:

   l) Configure GCM_CCMPH bits to '10' in CRYP_CR.

   m) Select the algorithm direction (encryption or decryption) by using the ALGODIR bit in CRYP_CR.

   n) Program the payload message into the CRYP_DIN register, and use the IFNF flag to determine if the input FIFO can receive data. Alternatively, the data could be programmed into the CRYP_DIN register by blocks of 8 words and the IFEM flag used to determine if the input FIFO can receive data (IFEM='1'). In parallel, the OFNE/OFFU flag of the CRYP_DOUT register can be monitored to check if the output FIFO is not empty.

   o) Repeat the previous step until all payload blocks have been encrypted or decrypted. Alternatively, DMA could be used.

4. CCM final phase

   This step generates the authentication tag. During this phase, the authentication tag of the message is generated and stored in the CRYP_DOUT register.

   p) Configure GCM_CCMPH[1:0] bits to '11' in CRYP_CR.

   q) Load the A0 initialized counter, and program the 128-bit A0 value by writing 4 times 32 bits into the CRYP_DIN register.

   r) Wait till the OFNE flag (FIFO output not empty) is set to '1' in the CRYP_SR register.

   s) Read the CRYP_DOUT register 4 times: the output corresponds to the encrypted authentication tag.

   t) Disable the cryptographic processor (CRYPEN bit in CRYP_CR = '0')

Note: The hardware does not perform the formatting of the original B0 and B1 packets and the tag comparison between encryption and decryption. They have to be handled by software.

The cryptographic *processor does not need to be disabled/enabled when moving from the header phase to the tag phase.*

AES cipher message authentication code (CMAC)

The CMAC algorithm allows authenticating the plaintext, and generating the corresponding tag. The CMAC sequence is identical to the CCM one, except that the payload phase is skipped.

23.3.3 Data type

Data enter the CRYP processor 32 bits (word) at a time as they are written into the CRYP_DIN register. The principle of the DES is that streams of data are processed 64 bits by 64 bits and, for each 64-bit block, the bits are numbered from M1 to M64, with M1 the left-most bit and M64 the right-most bit of the block. The same principle is used for the AES, but with a 128-bit block size.

The system memory organization is little-endian: whatever the data type (bit, byte, 16-bit half-word, 32-bit word) used, the least-significant data occupy the lowest address locations. A bit, byte, or half-word swapping operation (depending on the kind of data to be encrypted) therefore has to be performed on the data read from the IN FIFO before they enter the CRYP processor. The same swapping operation should be performed on the CRYP data before they are written into the OUT FIFO. For example, the operation would be byte swapping for an ASCII text stream.

The kind of data to be processed is configured with the DATATYPE bitfield in the CRYP control register (CRYP_CR).

Table 113. Data types

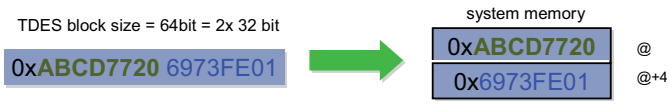
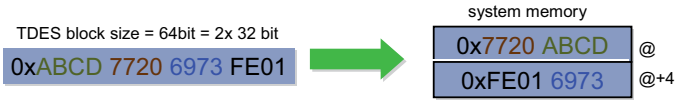
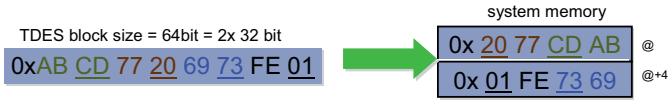
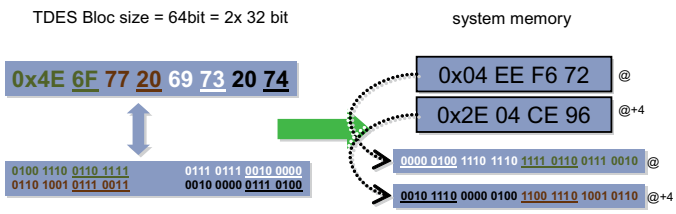
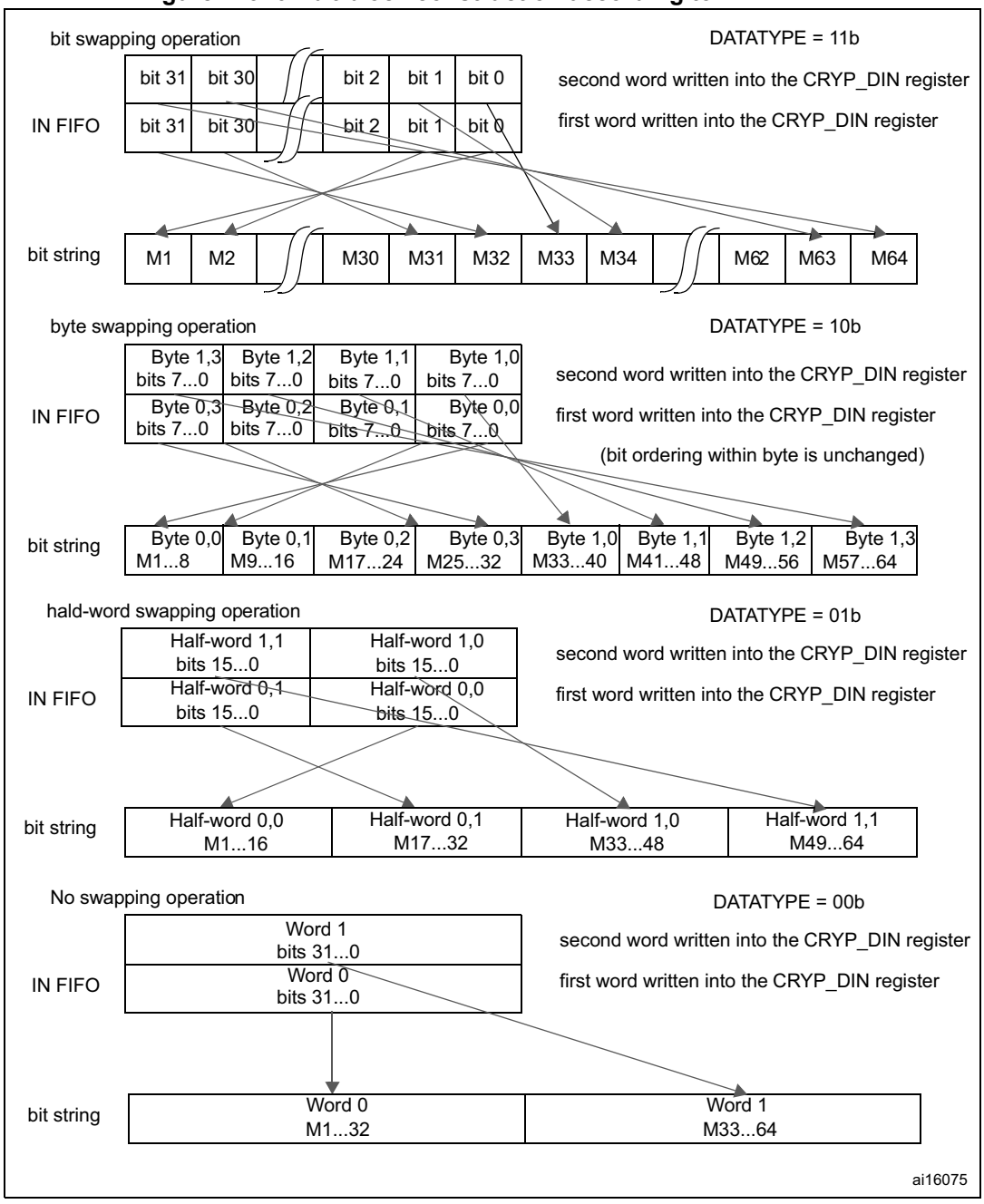
DATATYPE in CRYP_CR	Swapping performed	System memory data (plaintext or cypher)
00b	No swapping	Example: TDES block value **0xABCD77206973FE01** is represented in system memory as: 
01b	Half-word (16-bit) swapping	Example: TDES block value **0xABCD77206973FE01** is represented in system memory as: 
10b	Byte (8-bit) swapping	Example: TDES block value **0xABCD77206973FE01** is represented in system memory as: 
11b	Bit swapping	TDES block value **0x4E6F772069732074** is represented in system memory as: 

Figure 229 shows how the 64-bit data block M1...64 is constructed from two consecutive 32-bit words popped off the IN FIFO by the CRYP processor, according to the DATATYPE value. The same schematic can easily be extended to form the 128-bit block for the AES

cryptographic algorithm (for the AES, the block length is four 32-bit words, but swapping only takes place at word level, so it is identical to the one described here for the TDES).

Note: The same swapping is performed between the IN FIFO and the CRYP data block, and between the CRYP data block and the OUT FIFO.

Figure 229. 64-bit block construction according to DATATYPE



23.3.4 Initialization vectors - CRYP_IV0...1(L/R)

Initialization vectors are considered as two 64-bit data items. They therefore do not have the same data format and representation in system memory as plaintext or cypher data, and they are not affected by the DATATYPE value.

Initialization vectors are defined by two consecutive 32-bit words, CRYP_IVL (left part, noted as bits IV1...32) and CRYP_IVR (right part, noted as bits IV33...64).

During the DES or TDES CBC encryption, the CRYP_IV0(L/R) bits are XORed with the 64-bit data block popped off the IN FIFO after swapping (according to the DATATYPE value), that is, with the M1...64 bits of the data block. When the output of the DEA3 block is available, it is copied back into the CRYP_IV0(L/R) vector, and this new content is XORed with the next 64-bit data block popped off the IN FIFO, and so on.

During the DES or TDES CBC decryption, the CRYP_IV0(L/R) bits are XORed with the 64-bit data block (that is, with the M1...64 bits) delivered by the TDEA1 block before swapping (according to the DATATYPE value), and pushed into the OUT FIFO. When the XORed result is swapped and pushed into the OUT FIFO, the CRYP_IV0(L/R) value is replaced by the output of the IN FIFO, then the IN FIFO is popped, and a new 64-bit data block can be processed.

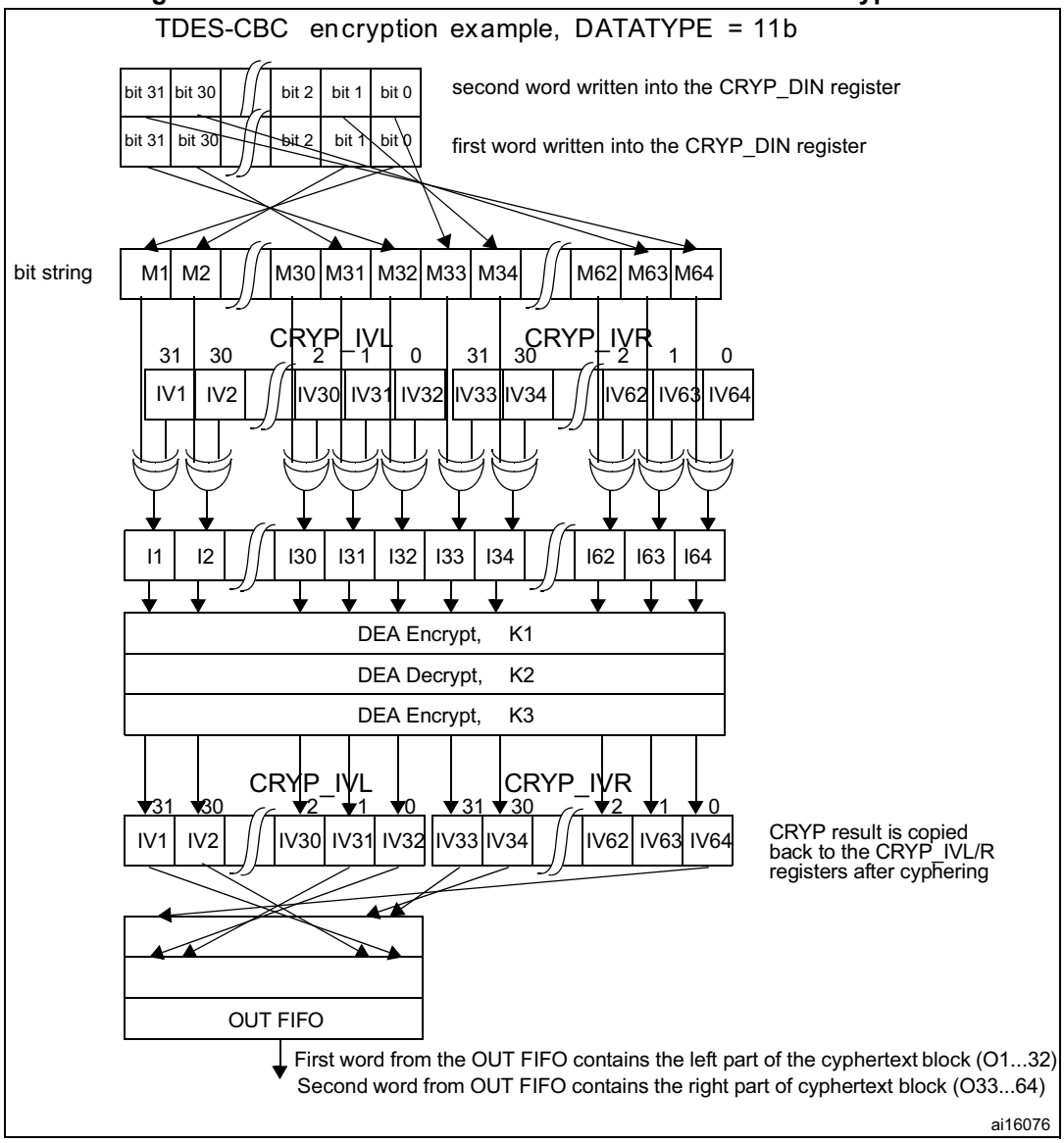
During the AES CBC encryption, the CRYP_IV0...1(L/R) bits are XORed with the 128-bit data block popped off the IN FIFO after swapping (according to the DATATYPE value). When the output of the AES core is available, it is copied back into the CRYP_IV0...1(L/R) vector, and this new content is XORed with the next 128-bit data block popped off the IN FIFO, and so on.

During the AES CBC decryption, the CRYP_IV0...1(L/R) bits are XORed with the 128-bit data block delivered by the AES core before swapping (according to the DATATYPE value) and pushed into the OUT FIFO. When the XORed result is swapped and pushed into the OUT FIFO, the CRYP_IV0...1(L/R) value is replaced by the output of the IN FIFO, then the IN FIFO is popped, and a new 128-bit data block can be processed.

During the AES CTR encryption or decryption, the CRYP_IV0...1(L/R) bits are encrypted by the AES core. Then the result of the encryption is XORed with the 128-bit data block popped off the IN FIFO after swapping (according to the DATATYPE value). When the XORed result is swapped and pushed into the OUT FIFO, the counter part of the CRYP_IV0...1(L/R) value (32 LSB) is incremented.

Any write operation to the CRYP_IV0...1(L/R) registers when bit BUSY = 1b in the CRYP_SR register is disregarded (CRYP_IV0...1(L/R) register content not modified). Thus, you must check that bit BUSY = 0b before modifying initialization vectors.

Figure 230. Initialization vectors use in the TDES-CBC encryption



23.3.5 CRYP busy state

When there is enough data in the input FIFO (at least 2 words for the DES or TDES algorithm mode, 4 words for the AES algorithm mode) and enough free-space in the output FIFO (at least 2 (DES/TDES) or 4 (AES) word locations), and when the bit CRYPEN = 1 in the CRYP_CR register, then the cryptographic processor automatically starts an encryption or decryption process (according to the value of the ALGODIR bit in the CRYP_CR register).

This process takes 48 AHB2 clock cycles for the Triple-DES algorithm, 16 AHB2 clock cycles for the simple DES algorithm, and 14, 16 or 18 AHB2 clock cycles for the AES with key lengths of 128, 192 or 256 bits, respectively. During the whole process, the BUSY bit in the CRYP_SR register is set to ‘1’. At the end of the process, two (DES/TDES) or four (AES) words are written by the CRYP Core into the output FIFO, and the BUSY bit is cleared. In

the CBC, CTR mode, the initialization vectors CRYP_IVx(L/R)R (x = 0..3) are updated as well.

A write operation to the key registers (CRYP_Kx(L/R)R, x = 0..3), the initialization registers (CRYP_IVx(L/R)R, x = 0..3), or to bits [9:2] in the CRYP_CR register are ignored when the cryptographic processor is busy (bit BUSY = 1b in the CRYP_SR register), and the registers are not modified. It is thus not possible to modify the configuration of the cryptographic processor while it is processing a block of data. It is however possible to clear the CRYPEN bit while BUSY = 1, in which case the ongoing DES, TDES or AES processing is completed and the two/four word results are written into the output FIFO, and then, only then, the BUSY bit is cleared.

Note: When a block is being processed in the DES or TDES mode, if the output FIFO becomes full and if the input FIFO contains at least one new block, then the new block is popped off the input FIFO and the BUSY bit remains high until there is enough space to store this new block into the output FIFO.

23.3.6 Procedure to perform an encryption or a decryption

Initialization

1. Initialize the peripheral (the order of operations is not important except for the key preparation for AES-ECB or AES-CBC decryption. The key size and the key value must be entered before preparing the key and the algorithm must be configured once the key has been prepared):
   a) Configure the key size (128-, 192- or 256-bit, in the AES only) with the KEYSIZE bits in the CRYP_CR register
   b) Write the symmetric key into the CRYP_KxL/R registers (2 to 8 registers to be written depending on the algorithm)
   c) Configure the data type (1-, 8-, 16- or 32-bit), with the DATATYPE bits in the CRYP_CR register
   d) In case of decryption in AES-ECB or AES-CBC, you must prepare the key: configure the key preparation mode by setting the ALGOMODE bits to ‘111’ in the CRYP_CR register. Then write the CRYPEN bit to ‘1’: the BUSY bit is set. Wait until BUSY returns to 0 (CRYPEN is automatically cleared as well): the key is prepared for decryption
   e) Configure the algorithm and chaining (the DES/TDES in ECB/CBC, the AES in ECB/CBC/CTR/GCM/CCM) with the ALGOMODE bits in the CRYP_CR register
   f) Configure the direction (encryption/decryption), with the ALGODIR bit in the CRYP_CR register
   g) Write the initialization vectors into the CRYP_IVxL/R register (in CBC or CTR modes only)
2. Flush the IN and OUT FIFOs by writing the FFLUSH bit to 1 in the CRYP_CR register

Processing when the DMA is used to transfer the data from/to the memory

1. Configure the DMA controller to transfer the input data from the memory. The transfer length is the length of the message. As message padding is not managed by the peripheral, the message length must be an entire number of blocks. The data are transferred in burst mode. The burst length is 4 words in the AES and 2 or 4 words in

the DES/TDES. The DMA should be configured to set an interrupt on transfer completion of the output data to indicate that the processing is finished.

2. Enable the cryptographic processor by writing the CRYPEN bit to 1. Enable the DMA requests by setting the DIEN and DOEN bits in the CRYP_DMACR register.
3. All the transfers and processing are managed by the DMA and the cryptographic processor. The DMA interrupt indicates that the processing is complete. Both FIFOs are normally empty and BUSY = 0.

Processing when the data are transferred by the CPU during interrupts

1. Enable the interrupts by setting the INIM and OUTIM bits in the CRYP_IMSCR register.
2. Enable the cryptographic processor by setting the CRYPEN bit in the CRYP_CR register.
3. In the interrupt managing the input data: load the input message into the IN FIFO. You can load 2 or 4 words at a time, or load data until the FIFO is full. When the last word of the message has been entered into the FIFO, disable the interrupt by clearing the INIM bit.
4. In the interrupt managing the output data: read the output message from the OUT FIFO. You can read 1 block (2 or 4 words) at a time or read data until the FIFO is empty. When the last word has been read, INIM=0, BUSY=0 and both FIFOs are empty (IFEM=1 and OFNE=0). You can disable the interrupt by clearing the OUTIM bit and, the peripheral by clearing the CRYPEN bit.

Processing without using the DMA nor interrupts

1. Enable the cryptographic processor by setting the CRYPEN bit in the CRYP_CR register.
2. Write the first blocks in the input FIFO (2 to 8 words).
3. Repeat the following sequence until the complete message has been processed:
   a) Wait for OFNE=1, then read the OUT-FIFO (1 block or until the FIFO is empty)
   b) Wait for IFNF=1, then write the IN FIFO (1 block or until the FIFO is full)
4. At the end of the processing, BUSY=0 and both FIFOs are empty (IFEM=1 and OFNE=0). You can disable the peripheral by clearing the CRYPEN bit.

23.3.7 Context swapping

If a context switching is needed because a new task launched by the OS requires this resource, the following tasks have to be performed for full context restoration (example when the DMA is used):

Case of the AES and DES

1. Context saving
   a) Stop DMA transfers on the IN FIFO by clearing the DIEN bit in the CRYP_DMACR register.
   b) Wait until both the IN and OUT FIFOs are empty (IFEM=1 and OFNE=0 in the CRYP_SR register) and the BUSY bit is cleared.
   c) Stop DMA transfers on the OUT FIFO by writing the DOEN bit to 0 in the CRYP_DMACR register and clear the CRYPEN bit.
   d) Save the current configuration (bits [9:2] and bits 19 in the CRYP_CR register) and, if not in ECB mode, the initialization vectors. The key value must already be available in the memory. When needed, save the DMA status (pointers for IN and OUT messages, number of remaining bytes, etc.).

      Additional bits should be saved when GCM/GMAC or CCM/CMAC algorithms are used:

   – bits [17:16] in the CRYP_CR register
   – context swap registers:

      CRYP_CSGCMCCM0..7 for GCM/GMAC or CCM/CMAC algorithm

      CRYP_CSGCM0..7 for GCM/GMAC algorithm.
2. Configure and execute the other processing.
3. Context restoration
   a) Configure the processor as in *Section 23.3.6: Procedure to perform an encryption or a decryption on page 744*, *Initialization* with the saved configuration. For the AES-ECB or AES-CBC decryption, the key must be prepared again.
   b) If needed, reconfigure the DMA controller to transfer the rest of the message.
   c) Enable the processor by setting the CRYPEN bit and, the DMA requests by setting the DIEN and DOEN bits.

Case of the TDES

Context swapping can be done in the TDES in the same way as in the AES. But as the input FIFO can contain up to 4 unprocessed blocks and as the processing duration per block is higher, it can be faster in certain cases to interrupt the processing without waiting for the IN FIFO to be empty.

1. Context saving
   a) Stop DMA transfers on the IN FIFO by clearing the DIEN bit in the CRYP_DMACR register.
   b) Disable the processor by clearing the CRYPEN bit (the processing will stop at the end of the current block).
   c) Wait until the OUT FIFO is empty (OFNE=0 in the CRYP_SR register) and the BUSY bit is cleared.
   d) Stop DMA transfers on the OUT FIFO by writing the DOEN bit to 0 in the CRYP_DMACR register.
   e) Save the current configuration (bits [9:2] and bits 19 in the CRYP_CR register) and, if not in ECB mode, the initialization vectors. The key value must already be available in the memory. When needed, save the DMA status (pointers for IN and OUT messages, number of remaining bytes, etc.). Read back the data loaded in

the IN FIFO that have not been processed and save them in the memory until the FIFO is empty.

Note: In GCM/GMAC or CCM/CMAC mode, bits [17:16] of the CRYP_CR register should also be saved.

2. Configure and execute the other processing.
3. Context restoration
   a) Configure the processor as in *Section 23.3.6: Procedure to perform an encryption or a decryption on page 744*, *Initialization* with the saved configuration. For the AES-ECB or AES-CBC decryption, the key must be prepared again.
   b) Write the data that were saved during context saving into the IN FIFO.
   c) If needed, reconfigure the DMA controller to transfer the rest of the message.
   d) Enable the processor by setting the CRYPEN bit and, the DMA requests by setting the DIEN and DOEN bits.

23.4 CRYP interrupts

There are two individual maskable interrupt sources generated by the CRYP. These two sources are combined into a single interrupt signal, which is the only interrupt signal from the CRYP that drives the NVIC (nested vectored interrupt controller). This combined interrupt, which is an OR function of the individual masked sources, is asserted if any of the individual interrupts listed below is asserted and enabled.

You can enable or disable the interrupt sources individually by changing the mask bits in the CRYP_IMSCR register. Setting the appropriate mask bit to ‘1’ enables the interrupt.

The status of the individual interrupt sources can be read either from the CRYP_RISR register, for raw interrupt status, or from the CRYP_MISR register, for the masked interrupt status.

Output FIFO service interrupt - OUTMIS

The output FIFO service interrupt is asserted when there is one or more (32-bit word) data items in the output FIFO. This interrupt is cleared by reading data from the output FIFO until there is no valid (32-bit) word left (that is, the interrupt follows the state of the OFNE (output FIFO not empty) flag).

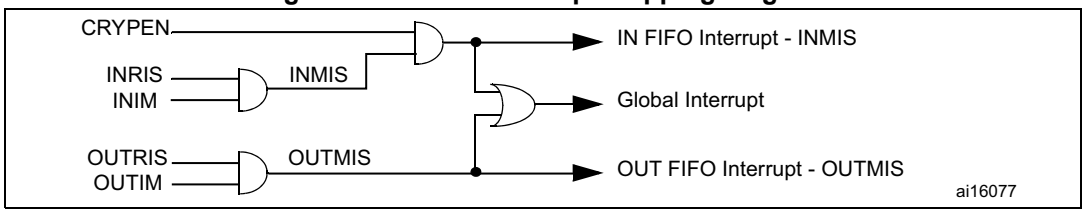
The output FIFO service interrupt OUTMIS is NOT enabled with the CRYP enable bit. Consequently, disabling the CRYP will not force the OUTMIS signal low if the output FIFO is not empty.

Input FIFO service interrupt - INMIS

The input FIFO service interrupt is asserted when there are less than four words in the input FIFO. It is cleared by performing write operations to the input FIFO until it holds four or more words.

The input FIFO service interrupt INMIS is enabled with the CRYP enable bit. Consequently, when CRYP is disabled, the INMIS signal is low even if the input FIFO is empty.

Figure 231. CRYP interrupt mapping diagram



23.5 CRYP DMA interface

The cryptographic processor provides an interface to connect to the DMA controller. The DMA operation is controlled through the CRYP DMA control register, CRYP_DMACR.

The burst and single transfer request signals are not mutually exclusive. They can both be asserted at the same time. For example, when there are 6 words available in the OUT FIFO, the burst transfer request and the single transfer request are asserted. After a burst transfer of 4 words, the single transfer request only is asserted to transfer the last 2 available words. This is useful for situations where the number of words left to be received in the stream is less than a burst.

Each request signal remains asserted until the relevant DMA clear signal is asserted. After the request clear signal is deasserted, a request signal can become active again, depending on the above described conditions. All request signals are deasserted if the CRYP peripheral is disabled or the DMA enable bit is cleared (DIEN bit for the IN FIFO and DOEN bit for the OUT FIFO in the CRYP_DMACR register).

Note: The DMA controller must be configured to perform burst of 4 words or less. Otherwise some data could be lost.

In order to let the DMA controller empty the OUT FIFO before filling up the IN FIFO, the OUTDMA channel should have a higher priority than the INDMA channel.

23.6 CRYP registers

The cryptographic core is associated with several control and status registers, eight key registers and four initialization vectors registers.

23.6.1 CRYP control register (CRYP_CR) for STM32F415/417xx

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRYPEN	FFLUSH	Reserved				KEYSIZE		DATATYPE		ALGOMODE[2:0]			ALGODIR	Res.	Res.
rw	w					rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:18 Reserved, must be kept at reset value

Bits 17:16 Reserved, must be kept at reset value

Bit 15 **CRYPEN:** Cryptographic processor enable

0: CRYP processor is disabled
1: CRYP processor is enabled

Note: The CRYPEN bit is automatically cleared by hardware when the key preparation process ends (ALGOMODE=111b) or GCM_CCM init Phase

Bit 14 **FFLUSH:** FIFO flush

When CRYPEN = 0, writing this bit to 1 flushes the IN and OUT FIFOs (that is read and write pointers of the FIFOs are reset. Writing this bit to 0 has no effect.
When CRYPEN = 1, writing this bit to 0 or 1 has no effect.
Reading this bit always returns 0.

Bits 13:10 Reserved, must be kept at reset value

Bits 9:8 **KEYSIZE[1:0]:** Key size selection (AES mode only)

This bitfield defines the bit-length of the key used for the AES cryptographic core. This bitfield is ‘don’t care’ in the DES or TDES modes.
00: 128 bit key length
01: 192 bit key length
10: 256 bit key length
11: Reserved, do not use this value

Bits 7:6 **DATATYPE[1:0]:** Data type selection

This bitfield defines the format of data entered in the CRYP_DIN register (refer to *Section 23.3.3: Data type*).
00: 32-bit data. No swapping of each word. First word pushed into the IN FIFO (or popped off the OUT FIFO) forms bits 1...32 of the data block, the second word forms bits 33...64.
01: 16-bit data, or half-word. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 2 half-words, which are swapped with each other.
10: 8-bit data, or bytes. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 4 bytes, which are swapped with each other.
11: bit data, or bit-string. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 32 bits (1st bit of the string at position 0), which are swapped with each other.

Bits 5:3 **ALGOMODE[2:0]:** Algorithm mode

000: TDES-ECB (triple-DES Electronic codebook): no feedback between blocks of data. Initialization vectors (CRYP_IV0(L/R)) are not used, three key vectors (K1, K2, and K3) are used (K0 is not used).

001: TDES-CBC (triple-DES Cipher block chaining): output block is XORed with the subsequent input block before its entry into the algorithm. Initialization vectors (CRYP_IV0L/R) must be initialized, three key vectors (K1, K2, and K3) are used (K0 is not used).

010: DES-ECB (simple DES Electronic codebook): no feedback between blocks of data. Initialization vectors (CRYP_IV0L/R) are not used, only one key vector (K1) is used (K0, K2, K3 are not used).

011: DES-CBC (simple DES Cipher block chaining): output block is XORed with the subsequent input block before its entry into the algorithm. Initialization vectors (CRYP_IV0L/R) must be initialized. Only one key vector (K1) is used (K0, K2, K3 are not used).

100: AES-ECB (AES Electronic codebook): no feedback between blocks of data. Initialization vectors (CRYP_IV0L/R...1L/R) are not used. All four key vectors (K0...K3) are used.

101: AES-CBC (AES Cipher block chaining): output block is XORed with the subsequent input block before its entry into the algorithm. Initialization vectors (CRYP_IV0L/R...1L/R) must be initialized. All four key vectors (K0...K3) are used.

110: AES-CTR (AES counter mode): output block is XORed with the subsequent input block before its entry into the algorithm. Initialization vectors (CRYP_IV0L/R...1L/R) must be initialized. All four key vectors (K0...K3) are used. CTR decryption does not differ from CTR encryption, since the core always encrypts the current counter block to produce the key stream that will be XORed with the plaintext or cipher in input. Thus, ALGODIR is don't care when ALGOMODE = 110b, and the key must NOT be unrolled (prepared) for decryption.

111: AES key preparation for decryption mode. Writing this value when CRYPEN = 1 immediately starts an AES round for key preparation. The secret key must have previously been loaded into the K0...K3 registers. The BUSY bit in the CRYP_SR register is set during the key preparation. After key processing, the resulting key is copied back into the K0...K3 registers, and the BUSY bit is cleared.

Bit 2 **ALGODIR:** Algorithm direction

0: Encrypt
1: Decrypt

Bits 1:0 Reserved, must be kept at reset value

Note: Writing to the KEYSIZE, DATATYPE, ALGOMODE and ALGODIR bits while BUSY=1 has no effect. These bits can only be configured when BUSY=0.
The FFLUSH bit has to be set only when BUSY=0. If not, the FIFO is flushed, but the block being processed may be pushed into the output FIFO just after the flush operation, resulting in a nonempty FIFO condition.

23.6.2 CRYP control register (CRYP_CR) for STM32F415/417xx

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												ALGO MODE [3]	Res.	GCM_CCMPH	
												rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRYPEN	FFLUSH	Reserved				KEYSIZE		DATATYPE		ALGOMODE[2:0]			ALGODIR	Reserved	
rw	w					rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:20 Reserved, forced by hardware to 0.

Bit 18 Reserved, forced by hardware to 0.

Bits 17:16 GCM_CCMPH[1:0]: no effect if “GCM or CCM algorithm” is not set

00: GCM_CCM init Phase
01: GCM_CCM header phase
10: GCM_CCM payload phase
11: GCM_CCM final phase

Bit 15 **CRYPEN:** Cryptographic processor enable

0: CRYP processor is disabled
1: CRYP processor is enabled

Note: The CRYPEN bit is automatically cleared by hardware when the key preparation process ends (ALGOMODE=111b) or GCM_CCM init Phase

Bit 14 **FFLUSH:** FIFO flush

When CRYPEN = 0, writing this bit to 1 flushes the IN and OUT FIFOs (that is read and write pointers of the FIFOs are reset. Writing this bit to 0 has no effect.
When CRYPEN = 1, writing this bit to 0 or 1 has no effect.
Reading this bit always returns 0.

Bits 13:10 Reserved, forced by hardware to 0.

Bits 9:8 **KEYSIZE[1:0]:** Key size selection (AES mode only)

This bitfield defines the bit-length of the key used for the AES cryptographic core. This bitfield is ‘don’t care’ in the DES or TDES modes.
00: 128 bit key length
01: 192 bit key length
10: 256 bit key length
11: Reserved, do not use this value

Bits 7:6 **DATATYPE[1:0]:** Data type selection

This bitfield defines the format of data entered in the CRYP_DIN register (refer to *Section 23.3.3: Data type*).
00: 32-bit data. No swapping of each word. First word pushed into the IN FIFO (or popped off the OUT FIFO) forms bits 1...32 of the data block, the second word forms bits 33...64.
01: 16-bit data, or half-word. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 2 half-words, which are swapped with each other.
10: 8-bit data, or bytes. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 4 bytes, which are swapped with each other.
11: bit data, or bit-string. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 32 bits (1st bit of the string at position 0), which are swapped with each other.

Bits 19 and 5:3 **ALGOMODE[3:0]:** Algorithm mode

0000: TDES-ECB (triple-DES Electronic codebook): no feedback between blocks of data. Initialization vectors (CRYP_IV0(L/R)) are not used, three key vectors (K1, K2, and K3) are used (K0 is not used).

0001: TDES-CBC (triple-DES Cipher block chaining): output block is XORed with the subsequent input block before its entry into the algorithm. Initialization vectors (CRYP_IV0L/R) must be initialized, three key vectors (K1, K2, and K3) are used (K0 is not used).

0010: DES-ECB (simple DES Electronic codebook): no feedback between blocks of data. Initialization vectors (CRYP_IV0L/R) are not used, only one key vector (K1) is used (K0, K2, K3 are not used).

0011: DES-CBC (simple DES Cipher block chaining): output block is XORed with the subsequent input block before its entry into the algorithm. Initialization vectors (CRYP_IV0L/R) must be initialized. Only one key vector (K1) is used (K0, K2, K3 are not used).

0100: AES-ECB (AES Electronic codebook): no feedback between blocks of data. Initialization vectors (CRYP_IV0L/R...1L/R) are not used. All four key vectors (K0...K3) are used.

0101: AES-CBC (AES Cipher block chaining): output block is XORed with the subsequent input block before its entry into the algorithm. Initialization vectors (CRYP_IV0L/R...1L/R) must be initialized. All four key vectors (K0...K3) are used.

0110: AES-CTR (AES Counter mode): output block is XORed with the subsequent input block before its entry into the algorithm. Initialization vectors (CRYP_IV0L/R...1L/R) must be initialized. All four key vectors (K0...K3) are used. CTR decryption does not differ from CTR encryption, since the core always encrypts the current counter block to produce the key stream that will be XORed with the plaintext or cipher in input. Thus, ALGODIR is don't care when ALGOMODE = 110b, and the key must NOT be unrolled (prepared) for decryption.

0111: AES key preparation for decryption mode. Writing this value when CRYPEN = 1 immediately starts an AES round for key preparation. The secret key must have previously been loaded into the K0...K3 registers. The BUSY bit in the CRYP_SR register is set during the key preparation. After key processing, the resulting key is copied back into the K0...K3 registers, and the BUSY bit is cleared.

1000: Galois Counter Mode (GCM). This algorithm mode is also used for the GMAC algorithm.

1001: Counter with CBC-MAC (CCM). This algorithm mode is also used for the CMAC algorithm.

Bit 2 **ALGODIR:** Algorithm direction

0: Encrypt

1: Decrypt

Bits 1:0 Reserved, must be kept to 0.

Note: Writing to the KEYSIZE, DATATYPE, ALGOMODE and ALGODIR bits while BUSY=1 has no effect. These bits can only be configured when BUSY=0.
The FFLUSH bit has to be set only when BUSY=0. If not, the FIFO is flushed, but the block being processed may be pushed into the output FIFO just after the flush operation, resulting in a nonempty FIFO condition.

23.6.3 CRYP status register (CRYP_SR)

Address offset: 0x04

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											BUSY	OFFU	OFNE	IFNF	IFEM
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value

Bit 4 **BUSY:** Busy bit

0: The CRYP Core is not processing any data. The reason is either that:

– the CRYP core is disabled (CRYPEN=0 in the CRYP_CR register) and the last processing has completed, or

– The CRYP core is waiting for enough data in the input FIFO or enough free space in the output FIFO (that is in each case at least 2 words in the DES, 4 words in the AES).

1: The CRYP core is currently processing a block of data or a key preparation (for AES decryption).

Bit 3 **OFFU:** Output FIFO full

0: Output FIFO is not full
1: Output FIFO is full

Bit 2 **OFNE:** Output FIFO not empty

0: Output FIFO is empty
1: Output FIFO is not empty

Bit 1 **IFNF:** Input FIFO not full

0: Input FIFO is full
1: Input FIFO is not full

Bit 0 **IFEM:** Input FIFO empty

0: Input FIFO is not empty
1: Input FIFO is empty

23.6.4 CRYP data input register (CRYP_DIN)

Address offset: 0x08

Reset value: 0x0000 0000

The CRYP_DIN register is the data input register. It is 32-bit wide. It is used to enter up to four 64-bit (TDES) or two 128-bit (AES) plaintext (when encrypting) or ciphertext (when decrypting) blocks into the input FIFO, one 32-bit word at a time.

The first word written into the FIFO is the MSB of the input block. The LSB of the input block is written at the end. Disregarding the data swapping, this gives:

- In the DES/TDES modes: a block is a sequence of bits numbered from bit 1 (leftmost bit) to bit 64 (rightmost bit). Bit 1 corresponds to the MSB (bit 31) of the first word entered into the FIFO, bit 64 corresponds to the LSB (bit 0) of the second word entered into the FIFO.
- In the AES mode: a block is a sequence of bits numbered from 0 (leftmost bit) to 127 (rightmost bit). Bit 0 corresponds to the MSB (bit 31) of the first word written into the FIFO, bit 127 corresponds to the LSB (bit 0) of the 4th word written into the FIFO.

To fit different data sizes, the data written in the CRYP_DIN register can be swapped before being processed by configuring the DATATYPE bits in the CRYP_CR register. Refer to *Section 23.3.3: Data type on page 739* for more details.

When CRYP_DIN register is written to, the data are pushed into the input FIFO. When at least two 32-bit words in the DES/TDES mode (or four 32-bit words in the AES mode) have been pushed into the input FIFO, and when at least 2 words are free in the output FIFO, the CRYP engine starts an encrypting or decrypting process. This process takes two 32-bit words in the DES/TDES mode (or four 32-bit words in the AES mode) from the input FIFO and delivers two 32-bit words (or 4, respectively) to the output FIFO per process round.

When CRYP_DIN register is read:

- If CRYPEN = 0, the FIFO is popped, and then the data present in the Input FIFO are returned, from the oldest one (first reading) to the newest one (last reading). The IFEM flag must be checked before each read operation to make sure that the FIFO is not empty.
- if CRYPEN = 1, an undefined value is returned.

After the CRYP_DIN register has been read once or several times, the FIFO must be flushed by setting the FFLUSH bit prior to processing new data.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAIN															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAIN															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATAIN:** Data input

Read = returns Input FIFO content if CRYPEN = 0, else returns an undefined value.

Write = Input FIFO is written.

23.6.5 CRYP data output register (CRYP_DOUT)

Address offset: 0x0C

Reset value: 0x0000 0000

The CRYP_DOUT register is the data output register. It is read-only and 32-bit wide. It is used to retrieve up to four 64-bit (TDES mode) or two 128-bit (AES mode) ciphertext (when encrypting) or plaintext (when decrypting) blocks from the output FIFO, one 32-bit word at a time.

Like for the input data, the MSB of the output block is the first word read from the output FIFO. The LSB of the output block is read at the end. Disregarding data swapping, this gives:

- In the DES/TDES modes: Bit 1 (leftmost bit) corresponds to the MSB (bit 31) of the first word read from the FIFO, bit 64 (rightmost bit) corresponds to the LSB (bit 0) of the second word read from the FIFO.
- In the AES mode: Bit 0 (leftmost bit) corresponds to the MSB (bit 31) of the first word read from the FIFO, bit 127 (rightmost bit) corresponds to the LSB (bit 0) of the 4th word read from the FIFO.

To fit different data sizes, the data can be swapped after processing by configuring the DATATYPE bits in the CRYP_CR register. Refer to *Section 23.3.3: Data type on page 739* for more details.

When CRYP_DOUT register is read, the last data entered into the output FIFO (pointed to by the read pointer) is returned.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAOUT															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAOUT															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DATAOUT:** Data output
Read = returns output FIFO content.
Write = No effect.

23.6.6 CRYP DMA control register (CRYP_DMACR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														DOEN	DIEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value

Bit 1 **DOEN:** DMA output enable
0: DMA for outgoing data transfer is disabled
1: DMA for outgoing data transfer is enabled

Bit 0 **DIEN:** DMA input enable
0: DMA for incoming data transfer is disabled
1: DMA for incoming data transfer is enabled

23.6.7 CRYP interrupt mask set/clear register (CRYP_IMSCR)

Address offset: 0x14

Reset value: 0x0000 0000

The CRYP_IMSCR register is the interrupt mask set or clear register. It is a read/write register. On a read operation, this register gives the current value of the mask on the relevant interrupt. Writing 1 to the particular bit sets the mask, enabling the interrupt to be read. Writing 0 to this bit clears the corresponding mask. All the bits are cleared to 0 when the peripheral is reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														OUTIM	INIM
														rw	rw

Bits 31:2 Reserved, must be kept at reset value

Bit 1 **OUTIM:** Output FIFO service interrupt mask
0: Output FIFO service interrupt is masked
1: Output FIFO service interrupt is not masked

Bit 0 **INIM:** Input FIFO service interrupt mask
0: Input FIFO service interrupt is masked
1: Input FIFO service interrupt is not masked

23.6.8 CRYP raw interrupt status register (CRYP_RISR)

Address offset: 0x18

Reset value: 0x0000 0001

The CRYP_RISR register is the raw interrupt status register. It is a read-only register. On a read, this register gives the current raw status of the corresponding interrupt prior to masking. A write has no effect.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														OUTRIS	INRIS
														r	r

Bits 31:2 Reserved, must be kept at reset value

Bit 1 **OUTRIS:** Output FIFO service raw interrupt status

Gives the raw interrupt state prior to masking of the output FIFO service interrupt.

0: Raw interrupt not pending

1: Raw interrupt pending

Bit 0 **INRIS:** Input FIFO service raw interrupt status

Gives the raw interrupt state prior to masking of the Input FIFO service interrupt.

0: Raw interrupt not pending

1: Raw interrupt pending

23.6.9 CRYP masked interrupt status register (CRYP_MISR)

Address offset: 0x1C

Reset value: 0x0000 0000

The CRYP_MISR register is the masked interrupt status register. It is a read-only register. On a read, this register gives the current masked status of the corresponding interrupt prior to masking. A write has no effect.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														OUTMIS	INMIS
														r	r

Bits 31:2 Reserved, must be kept at reset value

Bit 1 **OUTMIS:** Output FIFO service masked interrupt status
Gives the interrupt state after masking of the output FIFO service interrupt.
0: Interrupt not pending
1: Interrupt pending

Bit 0 **INMIS:** Input FIFO service masked interrupt status
Gives the interrupt state after masking of the input FIFO service interrupt.
0: Interrupt not pending
1: Interrupt pending when CRYPEN = 1

23.6.10 CRYP key registers (CRYP_K0...3(L/R)R)

Address offset: 0x20 to 0x3C

Reset value: 0x0000 0000

These registers contain the cryptographic keys.

In the TDES mode, keys are 64-bit binary values (number from left to right, that is the leftmost bit is bit 1), named K1, K2 and K3 (K0 is not used), each key consists of 56 information bits and 8 parity bits. The parity bits are reserved for error detection purposes and are not used by the current block. Thus, bits 8, 16, 24, 32, 40, 48, 56 and 64 of each 64-bit key value Kx[1:64] are not used.

In the AES mode, the key is considered as a single 128-, 192- or 256-bit long bit sequence, $k_0k_1k_2...k_{127/191/255}$ (k_0 being the leftmost bit). The AES key is entered into the registers as follows:

- for AES-128: $k_0..k_{127}$ corresponds to $b_{127}..b_0$ ($b_{255}..b_{128}$ are not used),
- for AES-192: $k_0..k_{191}$ corresponds to $b_{191}..b_0$ ($b_{255}..b_{192}$ are not used),
- for AES-256: $k_0..k_{255}$ corresponds to $b_{255}..b_0$.

In any case b_0 is the rightmost bit.

CRYP_K0LR (address offset: 0x20)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
b255	b254	b253	b252	b251	b250	b249	b248	b247	b246	b245	b244	b243	b242	b241	b240
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
b239	b238	b237	b236	b235	b234	b233	b232	b231	b230	b229	b228	b227	b226	b225	b224
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K0RR (address offset: 0x24)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
b223	b222	b221	b220	b219	b218	b217	b216	b215	b214	b213	b212	b211	b210	b209	b208
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
b207	b206	b205	b204	b203	b202	b201	b200	b199	b198	b197	b196	b195	b194	b193	b192
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K1LR (address offset: 0x28)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k1.1 b191	k1.2 b190	k1.3 b189	k1.4 b188	k1.5 b187	k1.6 b186	k1.7 b185	k1.8 b184	k1.9 b183	k1.10 b182	k1.11 b181	k1.12 b180	k1.13 b179	k1.14 b178	k1.15 b177	k1.16 b176
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
k1.17 b175	k1.18 b174	k1.19 b173	k1.20 b172	k1.21 b171	k1.22 b170	k1.23 b169	k1.24 b168	k1.25 b167	k1.26 b166	k1.27 b165	k1.28 b164	k1.29 b163	k1.30 b162	k1.31 b161	k1.32 b160
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K1RR (address offset: 0x2C)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k1.33 b159	k1.34 b158	k1.35 b157	k1.36 b156	k1.37 b155	k1.38 b154	k1.39 b153	k1.40 b152	k1.41 b151	k1.42 b150	k1.43 b149	k1.44 b148	k1.45 b147	k1.46 b146	k1.47 b145	k1.48 b144
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
k1.49 b143	k1.50 b142	k1.51 b141	k1.52 b140	k1.53 b139	k1.54 b138	k1.55 b137	k1.56 b136	k1.57 b135	k1.58 b134	k1.59 b133	k1.60 b132	k1.61 b131	k1.62 b130	k1.63 b129	k1.64 b128
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K2LR (address offset: 0x30)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k2.1 b127	k2.2 b126	k2.3 b125	k2.4 b124	k2.5 b123	k2.6 b122	k2.7 b121	k2.8 b120	k2.9 b119	k2.10 b118	k2.11 b117	k2.12 b116	k2.13 b115	k2.14 b114	k2.15 b113	k2.16 b112
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
k2.17 b111	k2.18 b110	k2.19 b109	k2.20 b108	k2.21 b107	k2.22 b106	k2.23 b105	k2.24 b104	k2.25 b103	k2.26 b102	k2.27 b101	k2.28 b100	k2.29 b99	k2.30 b98	k2.31 b97	k2.32 b96
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K2RR (address offset: 0x34)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k2.33 b95	k2.34 b94	k2.35 b93	k2.36 b92	k2.37 b91	k2.38 b90	k2.39 b89	k2.40 b88	k2.41 b87	k2.42 b86	k2.43 b85	k2.44 b84	k2.45 b83	k2.46 b82	k2.47 b81	k2.48 b80
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
k2.49 b79	k2.50 b78	k2.51 b77	k2.52 b76	k2.53 b75	k2.54 b74	k2.55 b73	k2.56 b72	k2.57 b71	k2.58 b70	k2.59 b69	k2.60 b68	k2.61 b67	k2.62 b66	k2.63 b65	k2.64 b64
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K3LR (address offset: 0x38)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k3.1 b63	k3.2 b62	k3.3 b61	k3.4 b60	k3.5 b59	k3.6 b58	k3.7 b57	k3.8 b56	k3.9 b55	k3.10 b54	k3.11 b53	k3.12 b52	k3.13 b51	k3.14 b50	k3.15 b49	k3.16 b48
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
k3.17 b47	k3.18 b46	k3.19 b45	k3.20 b44	k3.21 b43	k3.22 b42	k3.23 b41	k3.24 b40	k3.25 b39	k3.26 b38	k3.27 b37	k3.28 b36	k3.29 b35	k3.30 b34	k3.31 b33	k3.32 b32
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K3RR (address offset: 0x3C)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k3.33 b31	k3.34 b30	k3.35 b29	k3.36 b28	k3.37 b27	k3.38 b26	k3.39 b25	k3.40 b24	k3.41 b23	k3.42 b22	k3.43 b21	k3.44 b20	k3.45 b19	k3.46 b18	k3.47 b17	k3.48 b16
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
k3.49 b15	k3.50 b14	k3.51 b13	k3.52 b12	k3.53 b11	k3.54 b10	k3.55 b9	k3.56 b8	k3.57 b7	k3.58 b6	k3.59 b5	k3.60 b4	k3.61 b3	k3.62 b2	k3.63 b1	k3.64 b0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Note: Write accesses to these registers are disregarded when the cryptographic processor is busy (bit BUSY = 1 in the CRYP_SR register).

23.6.11 CRYP initialization vector registers (CRYP_IV0...1(L/R)R)

Address offset: 0x40 to 0x4C

Reset value: 0x0000 0000

The CRYP_IV0...1(L/R)R are the left-word and right-word registers for the initialization vector (64 bits for DES/TDES and 128 bits for AES) and are used in the CBC (Cipher block chaining) and Counter (CTR) modes. After each computation round of the TDES or AES Core, the CRYP_IV0...1(L/R)R registers are updated as described in *Section : DES and TDES Cipher block chaining (DES/TDES-CBC) mode on page 726*, *Section : AES Cipher block chaining (AES-CBC) mode on page 730* and *Section : AES counter mode (AES-CTR) mode on page 732*.

IV0 is the leftmost bit whereas IV63 (DES, TDES) or IV127 (AES) are the rightmost bits of the initialization vector. IV1(L/R)R is used only in the AES.

CRYP_IV0LR (address offset: 0x40)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV0	IV1	IV2	IV3	IV4	IV5	IV6	IV7	IV8	IV9	IV10	IV11	IV12	IV13	IV14	IV15
rw	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
IV16	IV17	IV18	IV19	IV20	IV21	IV22	IV23	IV24	IV25	IV26	IV27	IV28	IV29	IV30	IV31
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

CRYP_IV0RR (address offset: 0x44)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV32	IV33	IV34	IV35	IV36	IV37	IV38	IV39	IV40	IV41	IV42	IV43	IV44	IV45	IV46	IV47
rw	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
IV48	IV49	IV50	IV51	IV52	IV53	IV54	IV55	IV56	IV57	IV58	IV59	IV60	IV61	IV62	IV63
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

CRYP_IV1LR (address offset: 0x48)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV64	IV65	IV66	IV67	IV68	IV69	IV70	IV71	IV72	IV73	IV74	IV75	IV76	IV77	IV78	IV79
rw	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
IV80	IV81	IV82	IV83	IV84	IV85	IV86	IV87	IV88	IV89	IV90	IV91	IV92	IV93	IV94	IV95
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

CRYP_IV1RR (address offset: 0x4C)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV96	IV97	IV98	IV99	IV100	IV101	IV102	IV103	IV104	IV105	IV106	IV107	IV108	IV109	IV110	IV111
rw	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
IV112	IV113	IV114	IV115	IV116	IV117	IV118	IV119	IV120	IV121	IV122	IV123	IV124	IV125	IV126	IV127
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: In DES/3DES modes, only CRYP_IV0(L/R) is used.

Write access to these registers are disregarded when the cryptographic processor is busy (bit BUSY = 1 in the CRYP_SR register).

23.6.12 CRYP context swap registers (CRYP_CSGCMCCM0..7R and CRYP_CSGCM0..7R) for STM32F42xxx and STM32F43xxx

Address offset:

- CRYP_CSGCMCCM0..7: 0x050 to 0x06C: used for GCM/GMAC or CCM/CMAC alogrithm only
- CRYP_CSGCM0..7: 0x070 to 0x08C: used for GCM/GMAC algorithm only

Reset value: 0x0000 0000

These registers contain the complete internal register states of the CRYP processor when the GCM/GMAC or CCM/CMAC algorithm is selected. They are useful when a context swap has to be performed because a high-priority task needs the cryptographic processor while it is already in use by another task.

When such an event occurs, the CRYP_CSGCMCCM0..7R and CRYP_CSGCM0..7R (in GCM/GMAC mode) or CRYP_CSGCMCCM0..7R (in CCM/CMAC mode) registers have to be read and the values retrieved have to be saved in the system memory space. The cryptographic processor can then be used by the preemptive task, and when the cryptographic computation is complete, the saved context can be read from memory and written back into the corresponding context swap registers.

Note: These registers are used only when GCM/GMAC or CCM/CMAC algorithm mode is selected.

CRYP_CSGCMCCMxR: where x=[7:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRYP_CSGCMCCMxR															
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
CRYP_CSGCMCCMxR															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

CRYP_CSGCMxR: where x=[7:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRYP_CSGCMxR															
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
CRYP_CSGCMxR															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

23.6.13 CRYP register map

Table 114. CRYP register map and reset values for STM32F415/417xx

Offset	Register name and reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00 0x00	CRYP_CR	Reserved																CRYPEN	FFLUSH	Reserved				KEYSIZE		DATATYPE		ALOMODE[2:0]			ALGODIR	Res.	
	Reset value																	0	0					0	0	0	0	0	0	0	0		
0x04	CRYP_SR	Reserved																											BUSY	OFFU	OFNE	IFNF	IFEM
	Reset value																												0	0	0	1	1
0x08	CRYP_DIN	DATAIN																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	CRYP_DOUT	DATAOUT																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	CRYP_DMACR	Reserved																														DOEN	DIEN
	Reset value																															0	0
0x14	CRYP_IMSCR	Reserved																														OUTIM	ININ
	Reset value																															0	0
0x18	CRYP_RISR	Reserved																														OUTRIS	INRIS
	Reset value																															0	1
0x1C	CRYP_MISR	Reserved																														OUTMIS	IN%IS
	Reset value																															0	0
0x20	CRYP_K0LR	CRYP_K0LR																															
	Reset value	0	0	0	0	0	0	0	0	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	CRYP_K0RR	CRYP_K0RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x38	CRYP_K3LR	CRYP_K3LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	CRYP_K3RR	CRYP_K3RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	CRYP_IV0LR	CRYP_IV0LR																															
	Reset value	0	0	0	0	0	0	0	0	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 114. CRYP register map and reset values for STM32F415/417xx (continued)

Offset	Register name and 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
0x44	CRYP_IV0RR	CRYP_IV0RR																															
	Reset value	0	0	0	0	0	0	0	0	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	CRYP_IV1LR	CRYP_IV1LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	CRYP_IV1RR	CRYP_IV1RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 115. CRYP register map and reset values for STM32F43xxx

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00 0x00	CRYP_CR	Reserved												ALGOMODE[3]	Res.	GCM_CCMPH		CRYPEN	FFLUSH	Reserved				KEYSIZE		DATATYPE		ALOMODE[2:0]			ALGODIR	Res..	
	Reset value															0	0	0	0					0	0	0	0	0	0	0	0		
0x04	CRYP_SR	Reserved																											BUSY	OFFU	OFNE	IFNF	IFEM
	Reset value																												0	0	0	1	1
0x08	CRYP_DIN	DATAIN																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	CRYP_DOUT	DATAOUT																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	CRYP_DMACR	Reserved																														DOEN	DIEN
	Reset value																															0	0
0x14	CRYP_IMSCR	Reserved																														OUTIM	INIM
	Reset value																															0	0
0x18	CRYP_RISR	Reserved																														OUTRIS	INRIS
	Reset value																															0	1

Table 115. CRYP register map and reset values for STM32F43xxx (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C	CRYP_MISR	Reserved																														OUTMIS	IN%IS
	Reset value																															0	0
0x20	CRYP_K0LR	CRYP_K0LR																															
	Reset value	0	0	0	0	0	0	0	0	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	CRYP_K0RR	CRYP_K0RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
...																																	
0x38	CRYP_K3LR	CRYP_K3LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	CRYP_K3RR	CRYP_K3RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	CRYP_IV0LR	CRYP_IV0LR																															
	Reset value	0	0	0	0	0	0	0	0	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	CRYP_IV0RR	CRYP_IV0RR																															
	Reset value	0	0	0	0	0	0	0	0	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	CRYP_IV1LR	CRYP_IV1LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	CRYP_IV1RR	CRYP_IV1RR																															
	Reset value	0	0	0	0	0	0	0	0	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	CRYP_CSGCMCCMR	CRYP_CSGCMCCM0R																															
	Reset value	0	0	0	0	0	0	0	0	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	CRYP_CSGCMCCM1R	CRYP_CSGCMCCM1R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x58	CRYP_CSGCMCCM2R	CRYP_CSGCMCCM2R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	CRYP_CSGCMCCM3R	CRYP_CSGCMCCM3R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 115. CRYP register map and reset values for STM32F43xxx (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x60	CRYP_CSGCMCCM4R	CRYP_CSGCMCCM4R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x64	CRYP_CSGCMCCM5R	CRYP_CSGCMCCM5R																															
	Reset value	0	0	0	0	0	0	0	0	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	CRYP_CSGCMCCM6R	CRYP_CSGCMCCM6R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6C	CRYP_CSGCMCCM7R	CRYP_CSGCMCCM7R																															
	Reset value	0	0	0	0	0	0	0	0	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	CRYP_CSGCM0R	CRYP_CSGCM0R																															
	Reset value	0	0	0	0	0	0	0	0	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	CRYP_CSGCM1R	CRYP_CSGCM1R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x78	CRYP_CSGCM2R	CRYP_CSGCM2R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7C	CRYP_CSGCM3R	CRYP_CSGCM3R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x80	CRYP_CSGCM4R	CRYP_CSGCM4R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	CRYP_CSGCM5R	CRYP_CSGCM5R																															
	Reset value	0	0	0	0	0	0	0	0	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	CRYP_CSGCM6R	CRYP_CSGCM6R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C	CRYP_CSGCM7R	CRYP_CSGCM7R																															
	Reset value	0	0	0	0	0	0	0	0	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 2.3: Memory map* for the register boundary addresses.

24 Random number generator (RNG)

This section applies to the whole STM32F4xx family, unless otherwise specified.

24.1 RNG introduction

The RNG processor is a random number generator, based on a continuous analog noise, that provides a random 32-bit value to the host when read.

The RNG passed the FIPS PUB 140-2 (2001 October 10) tests with a success ratio of 99%.

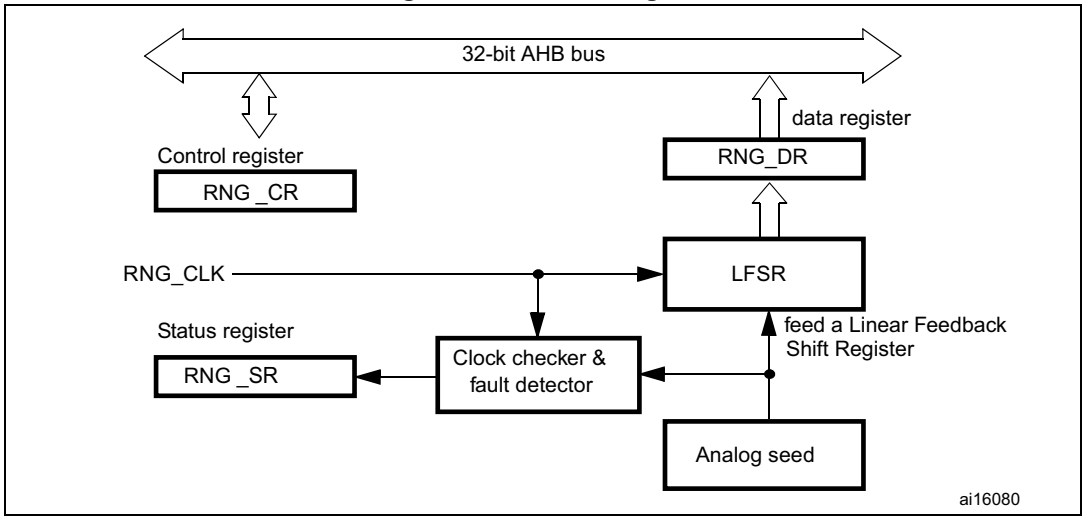
24.2 RNG main features

- It delivers 32-bit random numbers, produced by an analog generator
- 40 periods of the RNG_CLK clock signal between two consecutive random numbers
- Monitoring of the RNG entropy to flag abnormal behavior (generation of stable values, or of a stable sequence of values)
- It can be disabled to reduce power consumption

24.3 RNG functional description

Figure 232 shows the RNG block diagram.

Figure 232. Block diagram



1. For more details about RNG Clock (RNG_CLK) source, please refer to *Section 6: Reset and clock control for STM32F42xxx and STM32F43xxx (RCC)* and *Section 7: Reset and clock control for STM32F405xx/07xx and STM32F415xx/17xx(RCC)*.

The random number generator implements an analog circuit. This circuit generates seeds that feed a linear feedback shift register (RNG_LFSR) in order to produce 32-bit random numbers.

The analog circuit is made of several ring oscillators whose outputs are XORed to generate the seeds. The RNG_LFSR is clocked by a dedicated clock (RNG_CLK) at a constant frequency, so that the quality of the random number is independent of the HCLK frequency. The contents of the RNG_LFSR are transferred into the data register (RNG_DR) when a significant number of seeds have been introduced into the RNG_LFSR.

In parallel, the analog seed and the dedicated RNG_CLK clock are monitored. Status bits (in the RNG_SR register) indicate when an abnormal sequence occurs on the seed or when the frequency of the RNG_CLK clock is too low. An interrupt can be generated when an error is detected.

24.3.1 Operation

To run the RNG, follow the steps below:

1. Enable the interrupt if needed (to do so, set the IE bit in the RNG_CR register). An interrupt is generated when a random number is ready or when an error occurs.
2. Enable the random number generation by setting the RNGEN bit in the RNG_CR register. This activates the analog part, the RNG_LFSR and the error detector.
3. At each interrupt, check that no error occurred (the SEIS and CEIS bits should be ‘0’ in the RNG_SR register) and that a random number is ready (the DRDY bit is ‘1’ in the RNG_SR register). The contents of the RNG_DR register can then be read.

As required by the FIPS PUB (Federal Information Processing Standard Publication) 140-2, the first random number generated after setting the RNGEN bit should not be used, but saved for comparison with the next generated random number. Each subsequent generated random number has to be compared with the previously generated number. The test fails if any two compared numbers are equal (continuous random number generator test).

24.3.2 Error management

If the CEIS bit is read as ‘1’ (clock error)

In the case of a clock, the RNG is no more able to generate random numbers because the RNG_CLK clock is not correct. Check that the clock controller is correctly configured to provide the RNG clock and clear the CEIS bit. The RNG can work when the CECS bit is ‘0’. The clock error has no impact on the previously generated random numbers, and the RNG_DR register contents can be used.

If the SEIS bit is read as ‘1’ (seed error)

In the case of a seed error, the generation of random numbers is interrupted for as long as the SECS bit is ‘1’. If a number is available in the RNG_DR register, it must not be used because it may not have enough entropy.

What you should do is clear the SEIS bit, then clear and set the RNGEN bit to reinitialize and restart the RNG.

24.4 RNG registers

The RNG is associated with a control register, a data register and a status register. They have to be accessed by words (32 bits).

24.4.1 RNG control register (RNG_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												IE	RNGEN	Reserved	
												rw	rw		

Bits 31:4 Reserved, must be kept at reset value

Bit 3 **IE:** Interrupt enable
0: RNG Interrupt is disabled
1: RNG Interrupt is enabled. An interrupt is pending as soon as DRDY=1 or SEIS=1 or CEIS=1 in the RNG_SR register.

Bit 2 **RNGEN:** Random number generator enable
0: Random number generator is disabled
1: random Number Generator is enabled.

Bits 1:0 Reserved, must be kept at reset value

24.4.2 RNG status register (RNG_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									SEIS	CEIS	Reserved		SECS	CECS	DRDY
									rc_w0	rc_w0			r	r	r

Bits 31:3 Reserved, must be kept at reset value

Bit 6 **SEIS:** Seed error interrupt status
This bit is set at the same time as SECS, it is cleared by writing it to 0.
0: No faulty sequence detected
1: One of the following faulty sequences has been detected:
– More than 64 consecutive bits at the same value (0 or 1)
– More than 32 consecutive alternances of 0 and 1 (0101010101...01)
An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 5 **CEIS:** Clock error interrupt status
This bit is set at the same time as CECS, it is cleared by writing it to 0.
0: The RNG_CLK clock was correctly detected
1: The RNG_CLK was not correctly detected ($f_{RNG\_CLK} < f_{HCLK}/16$)
An interrupt is pending if IE = 1 in the RNG_CR register.

Bits 4:3 Reserved, must be kept at reset value

Bit 2 **SECS:** Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: One of the following faulty sequences has been detected:

– More than 64 consecutive bits at the same value (0 or 1)

– More than 32 consecutive alternances of 0 and 1 (0101010101...01)

Bit 1 **CECS:** Clock error current status

0: The RNG_CLK clock has been correctly detected. If the CEIS bit is set, this means that a clock error was detected and the situation has been recovered

1: The RNG_CLK was not correctly detected ($f_{RNG\_CLK} < f_{HCLK}/16$).

Bit 0 **DRDY:** Data ready

0: The RNG_DR register is not yet valid, no random data is available

1: The RNG_DR register contains valid random data

Note: An interrupt is pending if IE = 1 in the RNG_CR register.

Once the RNG_DR register has been read, this bit returns to 0 until a new valid value is computed.

24.4.3 RNG data register (RNG_DR)

Address offset: 0x08

Reset value: 0x0000 0000

The RNG_DR register is a read-only register that delivers a 32-bit random value when read. After being read, this register delivers a new random value after a maximum time of 40 periods of the RNG_CLK clock. The software must check that the DRDY bit is set before reading the RNDATA value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNDATA															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNDATA															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RNDATA:** Random data

32-bit random data.

24.4.4 RNG register map

Table 116 gives the RNG register map and reset values.

Table 116. RNG register map and reset map

Offset	Register name reset value	Register size																												
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RNG_CR 0x0000000	Reserved																												IE	RNGEN	Reserved	
0x04	RNG_SR 0x0000000	Reserved																									SEIS	CEIS	Reserved		SECS	CECS	DRDY
0x08	RNG_DR 0x0000000	RNDATA[31:0]																															

25 Hash processor (HASH)

This section applies to STM32F415/417xx and STM32F43xxx devices.

25.1 HASH 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. It computes a message digest (160 bits for the SHA-1 algorithm, 256 bits for the SHA-256 algorithm and 224 bits for the SHA-224 algorithm,128 bits for the MD5 algorithm) for messages of up to (2^{64} – 1) bits, while HMAC algorithms provide a way of authenticating messages by means of hash functions. HMAC algorithms consist in calling the SHA-1, SHA-224, SHA-256 or MD5 hash function twice.

25.2 HASH main features

- Suitable for data authentication applications, compliant with:
  - FIPS PUB 180-2 (Federal Information Processing Standards Publication 180-2)
  - Secure Hash Standard specifications (SHA-1, SHA-224 and SHA-256)
  - IETF RFC 1321 (Internet Engineering Task Force Request For Comments number 1321) specifications (MD5)
- Fast computation of SHA-1, SHA-224 and SHA-256, and MD5 (SHA-224 and SHA-256 are available on STM32F43xxx only)
- AHB slave peripheral
- 32-bit data words for input data, supporting word, half-word, byte and bit bit-string representations, with little-endian data representation only.
- Automatic swapping to comply with the big-endian SHA1, SHA-224 and SHA-256 computation standard with little-endian input bit-string representation
- Automatic padding to complete the input bit string to fit modulo 512 (16 × 32 bits) message digest computing
- 5× 32-bit words (H0 to H5) on STM32F415/417xx and 8 × 32-bit words (H0 to H7) on STM32F43xxx for output message digest, reload able to continue interrupted message digest computation.
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
- Automatic data flow control with support for direct memory access (DMA)

Note: *Padding, as defined in the SHA-1*, SHA-224 and SHA-256 *algorithm, consists in adding a bit at bx1 followed by N bits at bx0 to get a total length congruent to 448 modulo 512. After this, the message is completed with a 64-bit integer which is the binary representation of the original message length.*
For this hash processor, the quanta for entering the message is a 32-bit word, so an additional information must be provided at the end of the message entry, which is the number of valid bits in the last 32-bit word entered.

25.3 HASH functional description

Figure 1 shows the block diagram of the hash processor.

Figure 233. Block diagram for STM32F415/417xx

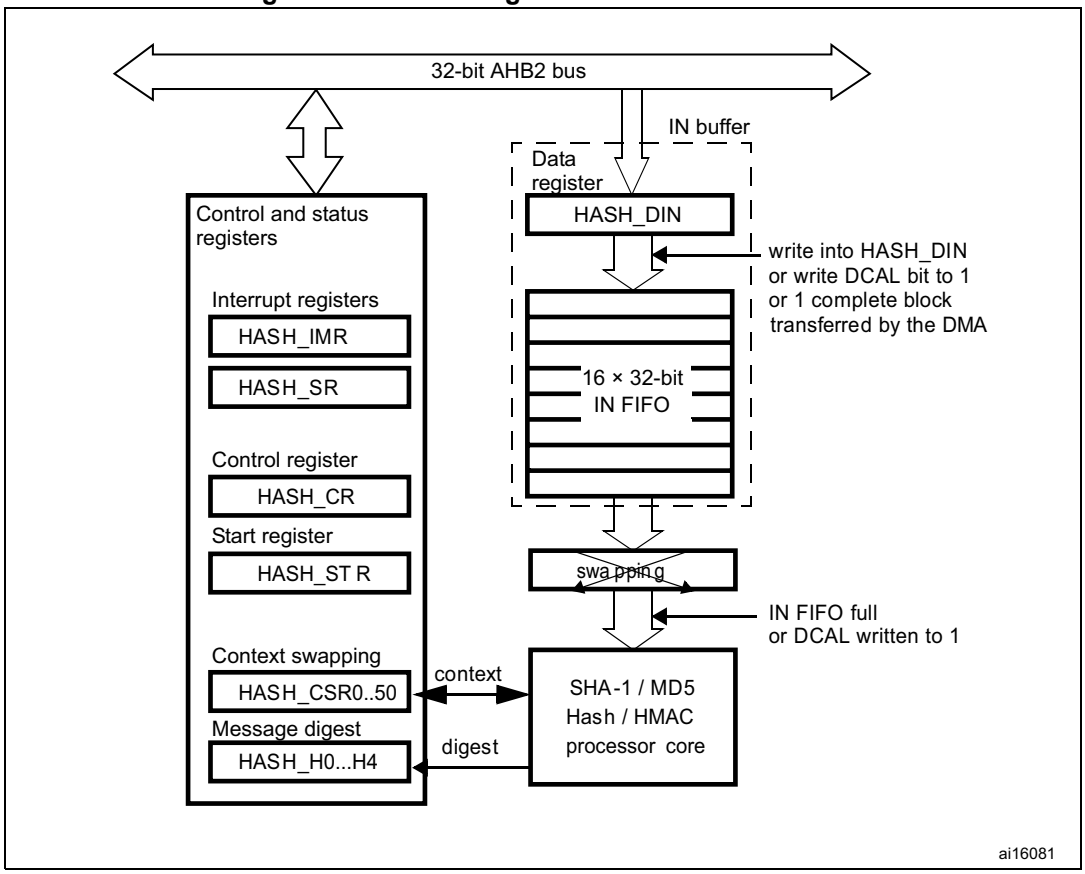
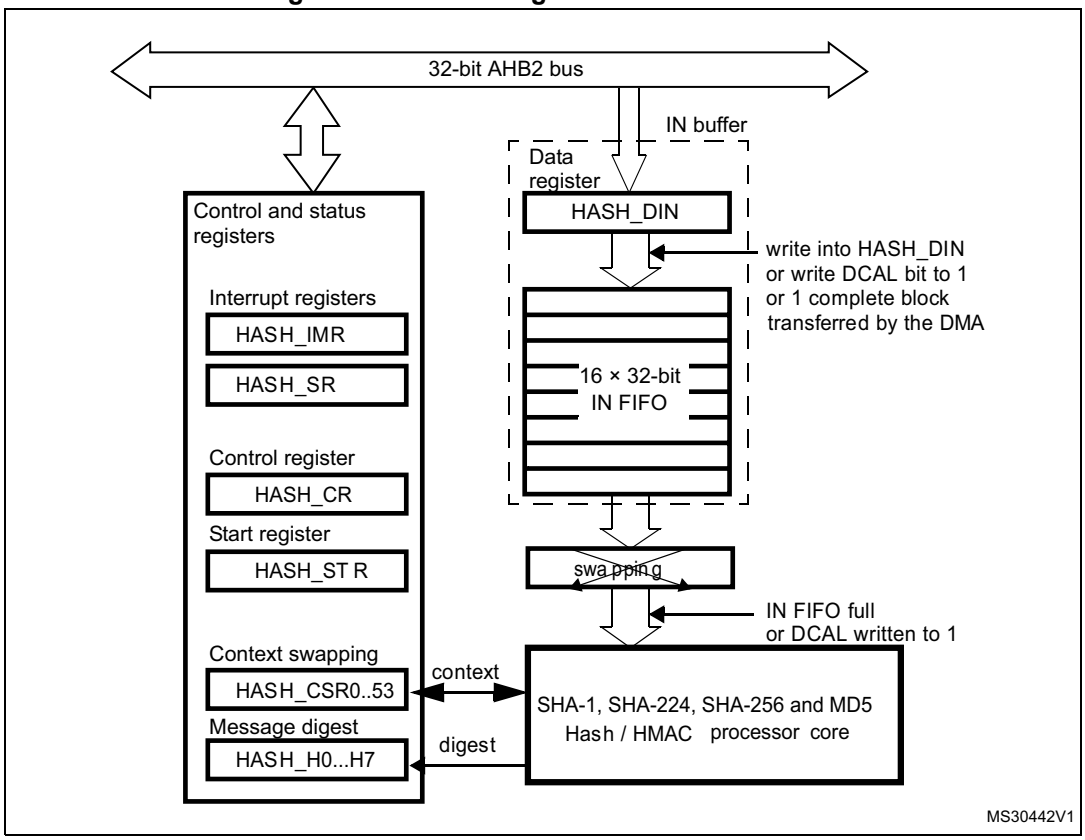


Figure 234. Block diagram for STM32F43xxx



The FIPS PUB 180-2 standard and the IETF RFC 1321 publication specify the SHA-1, SHA-224 and SHA-256 and MD5 secure hash algorithms, respectively, for computing a condensed representation of a message or data file. When a message of any length below 2^{64} bits is provided on input, the SHA-1, SHA-224 and SHA-256 and MD5 produce respective a 160-bit, 224 bit, 256 bit and 128-bit output string, respectively, called a message digest. The message digest can then be processed with a digital signature algorithm in order to generate or verify the signature for the message. Signing the message digest rather than the message often improves the efficiency of the process because the message digest is usually much smaller in size than the message. The verifier of a digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-1, SHA-224 and SHA-256 and MD5 are qualified as “secure” because it is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest. Any change to a message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify. For more detail on the SHA-1 or SHA-224 and SHA-256 algorithm, please refer to the FIPS PUB 180-2 (Federal Information Processing Standards Publication 180-2), 2002 august 1.

The current implementation of this standard works with little-endian input data convention. For example, the C string “abc” must be represented in memory as the 24-bit hexadecimal value 0x434241.

A message or data file to be processed by the hash processor should be considered a bit string. The length of the message is the number of bits in the message (the empty message

has length 0). You can consider that 32 bits of this bit string forms a 32-bit word. Note that the FIPS PUB 180-1 standard uses the convention that bit strings grow from left to right, and bits can be grouped as bytes (8 bits) or words (32 bits) (but some implementations also use half-words (16 bits), and implicitly, uses the big-endian byte (half-word) ordering. This convention is mainly important for padding (see *Section 1.3.4: Message padding on page 12*).

25.3.1 Duration of the processing

The computation of an intermediate block of a message takes:

- 66 HCLK clock cycles in SHA-1
- 50 HCLK clock cycles in SHA-224
- 50 HCLK clock cycles in SHA-256
- 50 HCLK clock cycles in MD5

to which you must add the time needed to load the 16 words of the block into the processor (at least 16 clock cycles for a 512-bit block).

The time needed to process the last block of a message (or of a key in HMAC) can be longer. This time depends on the length of the last block and the size of the key (in HMAC mode). Compared to the processing of an intermediate block, it can be increased by a factor of:

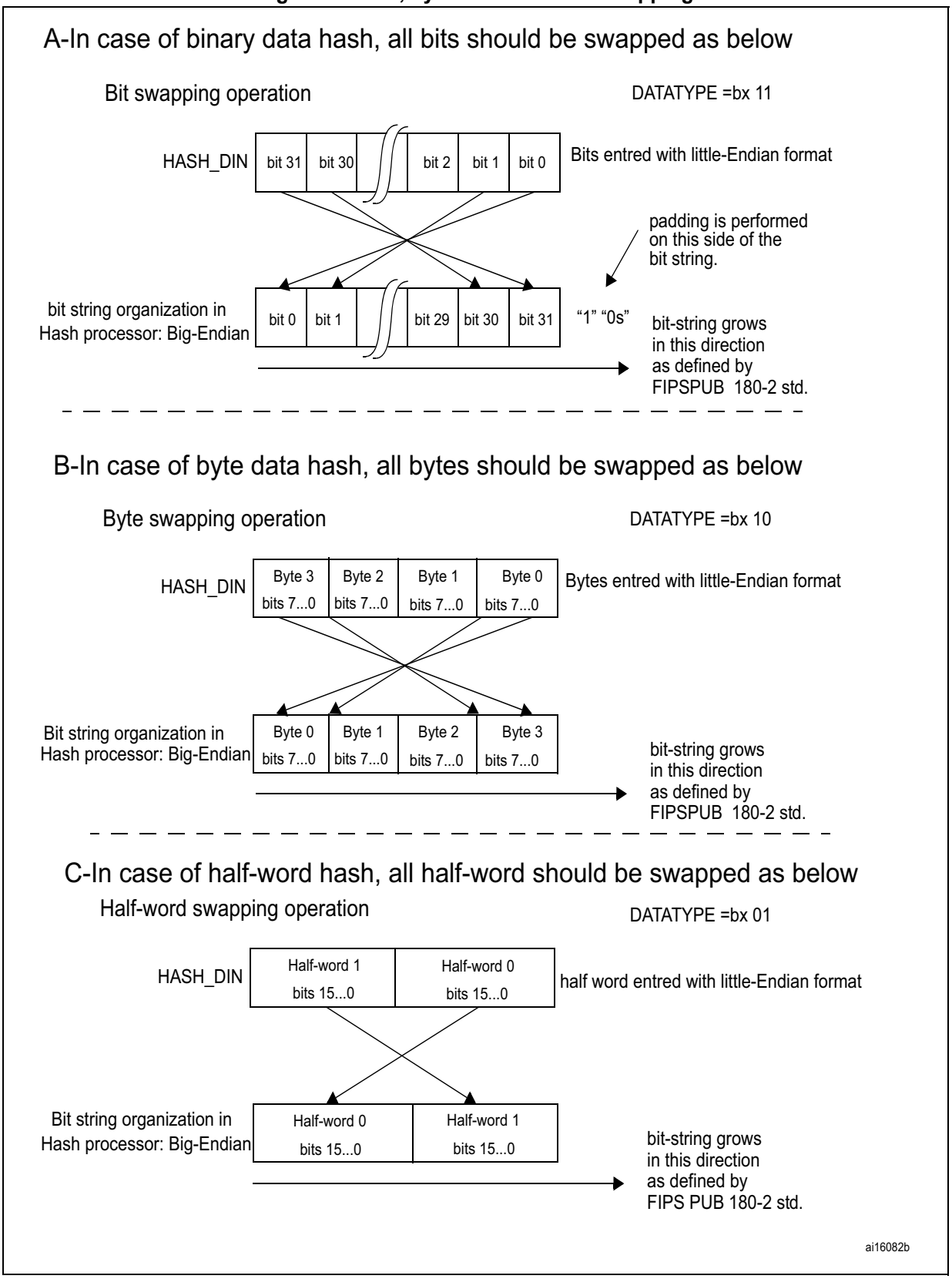
- 1 to 2.5 for a hash message
- around 2.5 for an HMAC input-key
- 1 to 2.5 for an HMAC message
- around 2.5 for an HMAC output key in case of a short key
- 3.5 to 5 for an HMAC output key in case of a long key

25.3.2 Data type

Data are entered into the hash processor 32 bits (word) at a time, by writing them into the HASH_DIN register. But the original bit-string can be organized in bytes, half-words or words, or even be represented as bits. As the system memory organization is little-endian and SHA1, SHA-224 and SHA-256 computation is big-endian, depending on the way the original bit string is grouped, a bit, byte, or half-word swapping operation is performed automatically by the hash processor.

The kind of data to be processed is configured with the DATATYPE bitfield in the HASH control register (HASH_CR).

Figure 235. Bit, byte and half-word swapping



The least significant bit of the message has to be at position 0 (right) in the first word entered into the hash processor, the 32nd bit of the bit string has to be at position 0 in the second word entered into the hash processor and so on.

25.3.3 Message digest computing

The HASH sequentially processes blocks of 512 bits when computing the message digest. Thus, each time 16 × 32-bit words (= 512 bits) have been written by the DMA or the CPU, into the hash processor, the HASH automatically starts computing the message digest. This operation is known as a partial digest computation.

The message to be processed is entered into the peripheral by 32-bit words written into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the input FIFO (IN FIFO) each time the register is written with new data. HASH_DIN and the input FIFO form a FIFO of a 17-word length (named the IN buffer).

The processing of a block can start only once the last value of the block has entered the IN FIFO. The peripheral must get the information as to whether the HASH_DIN register contains the last bits of the message or not. Two cases may occur:

- When the DMA is not used:
  - In case of a partial digest computation, this is done by writing an additional word into the HASH_DIN register (actually the first word of the next block). Then the software must wait until the processor is ready again (when DINIS=1) before writing new data into HASH_DIN.
  - In case of a final digest computation (last block entered), this is done by writing the DCAL bit to 1.
- When the DMA is used:

  The contents of the HASH_DIN register are interpreted automatically with the information sent by the DMA controller.
  - In case of a single DMA transfer: Multiple DMA transfer (MDMAT) bit should be cleared on STM32F43xxx. When the last block has been transferred to the HASH_DIN register via DMA channel, DCAL bit will be set to automatically to 1 in the HASH_STR register in order to launch the final digest calculation.
  - In case of a multiple DMA transfer (available only on STM32F43xxx): Multiple DMA transfer (MDMAT) bit should be set to 1 by software so DCAL bit does not get set automatically by HW, in this case the final digest calculation for hash and for each phases for HMAC (for more details about HMAC phases please refer to HMAC operation section) will not be launched a the end of the DMA transfer request, allowing the processor to receive a new DMA transfer. During the last DMA transfer, Multiple DMA transfer (MDMAT) bit should be cleared by software in order to set automatically DCAL bit at the end of the last bloc and lunch the final digest.
  - The contents of the HASH_DIN register are interpreted automatically with the information sent by the DMA controller.

This process —data entering + partial digest computation— continues until the last bits of the original message are written to the HASH_DIN register. As the length (number of bits) of a message can be any integer value, the last word written into the HASH processor may have a valid number of bits between 1 and 32. This number of valid bits in the last word, NBLW, has to be written into the HASH_STR register, so that message padding is correctly performed before the final message digest computation.

Once this is done, writing into HASH_STR with bit DCAL = 1 starts the processing of the last entered block of message by the hash processor. This processing consists in:

- Automatically performing the message padding operation: the purpose of this operation is to make the total length of a padded message a multiple of 512. The HASH sequentially processes blocks of 512 bits when computing the message digest
- Computing the final message digest

When the DMA is enabled, it provides the information to the hash processor when it is transferring the last data word. Then the padding and digest computation are performed automatically as if DCAL had been written to 1.

25.3.4 Message padding

Message padding consists in appending a “1” followed by m “0”s followed by a 64-bit integer to the end of the original message to produce a padded message block of length 512. The “1” is added to the last word written into the HASH_DIN register at the bit position defined by the NBLW bitfield, and the remaining upper bits are cleared (“0”s).

Example: let us assume that the original message is the ASCII binary-coded form of “abc”, of length L = 24:

```
byte 0    byte 1   byte 2   byte 3
01100001 01100010 01100011 UUUUUUUU
<-- 1st word written to HASH_DIN -->
```

NBLW has to be loaded with the value 24: a “1” is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH_DIN register (little-endian convention):

```
01100001 01100010 01100011 1UUUUUUU
```

Since L = 24, the number of bits in the above bit string is 25, and 423 “0”s are appended, making now 448. This gives (in hexadecimal, big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000
```

The L value, in two-word representation (that is 00000000 00000018) is appended. Hence the final padded message in hexadecimal:

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000028
```

If the HASH is programmed to use the little-endian byte input format, the above message has to be entered by doing the following steps:

1. 0xUU636261 is written into the HASH_DIN register (where ‘U’ means don’t care)
2. 0x18 is written into the HASH_STR register (the number of valid bits in the last word written into the HASH_DIN register is 24, as the original message length is 24 bits)
3. 0x10 is written into the HASH_STR register to start the message padding and digest computation. When NBLW ≠ 0x00, the message padding puts a “1” into the HASH_DIN register at the bit position defined by the NBLW value, and inserts “0”s at bit locations [31:(NBLW+1)]. When NBLW == 0x00, the message padding inserts one new word with

value 0x0000 0001. Then an all zero word (0x0000 0000) is added and the message length in a two-word representation, to get a block of 16 x 32-bit words.

4. The HASH computing is performed, and the message digest is then available in the HASH_Hx registers (x = 0...4) for the SHA-1 algorithm. For example:

```
H0 = 0xA9993E36
H1 = 0x4706816A
H2 = 0xBA3E2571
H3 = 0x7850C26C
H4 = 0x9CD0D89D
```

25.3.5 Hash operation

The hash function (SHA-1, SHA-224, SHA-256 and MD5) is selected when the INIT bit is written to '1' in the HASH_CR register while the MODE bit is at '0' in HASH_CR. The algorithm (SHA-1, SHA-224,SHA-256 or MD5) is selected at the same time (that is when the INIT bit is set) using the ALGO bits.

The message can then be sent by writing it word by word into the HASH_DIN register. When a block of 512 bits —that is 16 words— has been written, a partial digest computation starts upon writing the first data of the next block. The hash processor remains busy for 66 cycles for the SHA-1 algorithm, or 50 cycles for the MD5 algorithm, SHA-224 algorithm and SHA-256 algorithm.

The process can then be repeated until the last word of the message. If DMA transfers are used, refer to the *Procedure where the data are loaded by DMA* section. Otherwise, if the message length is not an exact multiple of 512 bits, then the HASH_STR register has to be written to launch the computation of the final digest.

Once computed, the digest can be read from the HASH_H0...HASH_H4 registers (for the MD5 algorithm, HASH_H4 is not relevant) on STM32F415/417xx, and from the HASH_H0...HASH_H7 registers on STM32F43xxx where:

HASH_H4..HASH_H7 are not relevant when the MD5 algorithm is selected,

HASH_H5.. HASH_H7 are not relevant when the SHA-1algorithm is selected,

HASH_H7 is not relevant when the SHA-224 algorithm is selected.

25.3.6 HMAC operation

The HMAC algorithm is used for message authentication, by irreversibly binding the message being processed to a key chosen by the user. For HMAC specifications, refer to "HMAC: keyed-hashing for message authentication, H. Krawczyk, M. Bellare, R. Canetti, February 1997.

Basically, the algorithm consists of two nested hash operations:

```
HMAC(message) = Hash[((key | pad) XOR 0x5C)
          | Hash(((key | pad) XOR 0x36) | message)]
```

where:

- `pad` is a sequence of zeroes needed to extend the key to the length of the underlying hash function data block (that is 512 bits for both the SHA-1, SHA224, SHA-256 and MD5 hash algorithms)
- | represents the concatenation operator

To compute the HMAC, four different phases are required:

1. The block is initialized by writing the INIT bit to ‘1’ with the MODE bit at ‘1’ and the ALGO bits set to the value corresponding to the desired algorithm. The LKEY bit must also be set during this phase if the key being used is longer than 64 bytes (in this case, the HMAC specifications specify that the hash of the key should be used in place of the real key).
2. The key (to be used for the inner hash function) is then given to the core. This operation follows the same mechanism as the one used to send the message in the hash operation (that is, by writing into HASH_DIN and, finally, into HASH_STR).
3. Once the last word has been entered and computation has started, the hash processor elaborates the key. It is then ready to accept the message text using the same mechanism as the one used to send the message in the hash operation.
4. After the first hash round, the hash processor returns “ready” to indicate that it is ready to receive the key to be used for the outer hash function (normally, this key is the same as the one used for the inner hash function). When the last word of the key is entered and computation starts, the HMAC result is made available in the HASH_H0...HASH_H4 registers on STM32F415/417xx and on HASH_H0...HASH_H7 registers on STM32F43xxx.

Note: 1 The computation latency of the HMAC primitive depends on the lengths of the keys and message. You could the HMAC as two nested underlying hash functions with the same key length (long or short).

25.3.7 Context swapping

It is possible to interrupt a hash/HMAC process to perform another processing with a higher priority, and to complete the interrupted process later on, when the higher-priority task is complete. To do so, the context of the interrupted task must be saved from the hash registers to memory, and then be restored from memory to the hash registers.

The procedures where the data flow is controlled by software or by DMA are described below.

Procedure where the data are loaded by software

The context can be saved only when no block is currently being processed. That is, you must wait for DINIS = 1 (the last block has been processed and the input FIFO is empty) or NBW ≠ 0 (the FIFO is not full and no processing is ongoing).

- Context saving:

  Store the contents of the following registers into memory:

  – HASH_IMR
  – HASH_STR
  – HASH_CR
  – HASH_CSR0 to HASH_CSR50 on STM32F415/417xx, and HASH_CSR0 to HASH_CSR53 on STM32F43xxx.

- Context restoring:

  The context can be restored when the high-priority task is complete. Please follow the order of the sequence below.

  a) Write the following registers with the values saved in memory: HASH_IMR, HASH_STR and HASH_CR
  b) Initialize the hash processor by setting the INIT bit in the HASH_CR register
  c) Write the HASH_CSR0 to HASH_CSR50 (STM32F415/417xx), and HASH_CSR0 to HASH_CSR53 (STM32F43xxx) registers with the values saved in memory

  You can now restart the processing from the point where it has been interrupted.

Procedure where the data are loaded by DMA

In this case it is not possible to predict if a DMA transfer is in progress or if the process is ongoing. Thus, you must stop the DMA transfers, then wait until the HASH is ready in order to interrupt the processing of a message.

- Interrupting a processing:
  – Clear the DMAE bit to disable the DMA interface
  – Wait until the current DMA transfer is complete (wait for DMAES = 0 in the HASH_SR register). Note that the block may or not have been totally transferred to the HASH.
  – Disable the corresponding channel in the DMA controller
  – Wait until the hash processor is ready (no block is being processed), that is wait for DINIS = 1
- The context saving and context restoring phases are the same as above (see *Procedure where the data are loaded by software*).

Reconfigure the DMA controller so that it transfers the end of the message. You can now restart the processing from the point where it was interrupted by setting the DMAE bit.

*Note: If context swapping does not involve HMAC operations, the HASH_CSR38 to HASH_CSR50 (*STM32F415/417xx*) and HASH_CSR38 to HASH_CSR53 (*STM32F43xxx*) registers do not have to be saved and restored.*

If context swapping occurs between two blocks (the last block was completely processed and the next block has not yet been pushed into the IN FIFO, NBW = 000 in the HASH_CR register), the HASH_CSR22 to HASH_CSR37 registers do not have to be saved and restored.

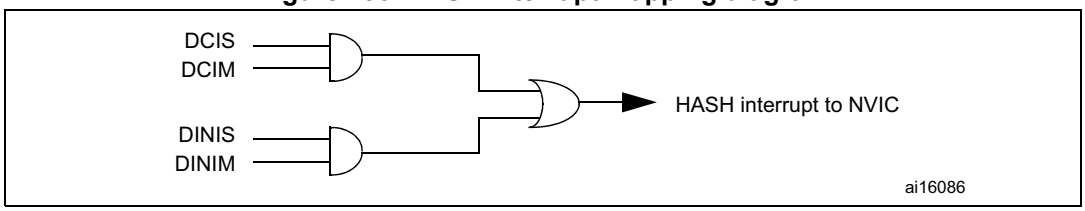
25.3.8 HASH interrupt

There are two individual maskable interrupt sources generated by the HASH processor. They are connected to the same interrupt vector.

You can enable or disable the interrupt sources individually by changing the mask bits in the HASH_IMR register. Setting the appropriate mask bit to 1 enables the interrupt.

The status of the individual interrupt sources can be read from the HASH_SR register.

Figure 236. HASH interrupt mapping diagram



25.4 HASH registers

The HASH core is associated with several control and status registers and five message digest registers.

All these registers are accessible through word accesses only, else an AHB error is generated.

25.4.1 HASH control register (HASH_CR) for STM32F415/417xx

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															LKEY
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DINNE	NBW				ALGO[0]	MODE	DATATYPE		DMAE	INIT	Reserved	
			r	r	r	r	r	rw	rw	rw	rw	rw	w		

Bits 31:17 Reserved, forced by hardware to 0.

Bit 16 **LKEY:** Long key selection
This bit selects between short key (≤64 bytes) or long key (> 64 bytes) in HMAC mode
0: Short key (≤64 bytes)
1: Long key (> 64 bytes)
Note: This selection is only taken into account when the INIT bit is set and MODE = 1. Changing this bit during a computation has no effect.

Bits 15:13 Reserved, forced by hardware to 0.

Bit 12 **DINNE:** DIN not empty

This bit is set when the HASH_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is written to 1.

0: No data are present in the data input buffer

1: The input buffer contains at least one word of data

Bits 11:8 **NBW:** Number of words already pushed

This bitfield reflects the number of words in the message that have already been pushed into the IN FIFO.

NBW increments (+1) when a write access is performed to the HASH_DIN register while DINNE = 1.

It goes to 0000 when the INIT bit is written to 1 or when a digest calculation starts (DCAL written to 1 or DMA end of transfer).

- If the DMA is not used:

0000 and DINNE=0: no word has been pushed into the DIN buffer (the buffer is empty, both the HASH_DIN register and the IN FIFO are empty)

0000 and DINNE=1: 1 word has been pushed into the DIN buffer (The HASH_DIN register contains 1 word, the IN FIFO is empty)

0001: 2 words have been pushed into the DIN buffer (the HASH_DIN register and the IN FIFO contain 1 word each)

...

1111: 16 words have been pushed into the DIN buffer

- If the DMA is used, NBW is the exact number of words that have been pushed into the IN FIFO.

Bit 7 **ALGO[1:0]:** Algorithm selection

These bits selects the SHA-1 or the MD5 algorithm:

0: SHA-1 algorithm selected

1: MD5 algorithm selected

Note: This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

Bit 6 **MODE:** Mode selection

This bit selects the HASH or HMAC mode for the selected algorithm:

0: Hash mode selected

1: HMAC mode selected. LKEY must be set if the key being used is longer than 64 bytes.

Note: This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

Bits 5:4 **DATATYPE:** Data type selection

Defines the format of the data entered into the HASH_DIN register:

00: 32-bit data. The data written into HASH_DIN are directly used by the HASH processing, without reordering.

01: 16-bit data, or half-word. The data written into HASH_DIN are considered as 2 half-words, and are swapped before being used by the HASH processing.

10: 8-bit data, or bytes. The data written into HASH_DIN are considered as 4 bytes, and are swapped before being used by the HASH processing.

11: bit data, or bit-string. The data written into HASH_DIN are considered as 32 bits (1st bit of the sting at position 0), and are swapped before being used by the HASH processing (1st bit of the string at position 31).

Bit 3 **DMAE:** DMA enable

0: DMA transfers disabled

1: DMA transfers enabled. A DMA request is sent as soon as the HASH core is ready to receive data.

Note: 1: This bit is cleared by hardware when the DMA asserts the DMA terminal count signal (while transferring the last data of the message). This bit is not cleared when the INIT bit is written to 1.

2: If this bit is written to 0 while a DMA transfer has already been requested to the DMA, DMAE is cleared but the current transfer is not aborted. Instead, the DMA interface remains internally enabled until the transfer is complete or INIT is written to 1.

Bit 2 **INIT:** Initialize message digest calculation

Writing this bit to 1 resets the hash processor core, so that the HASH is ready to compute the message digest of a new message.

Writing this bit to 0 has no effect.

Reading this bit always return 0.

Bits 1:0 Reserved, must be kept cleared.

25.4.2 HASH control register (HASH_CR) for STM32F43xxx

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ALGO[1]	Reserved	LKEY
													rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		MDMAT	DINNE	NBW				ALGO[0]	MODE	DATATYPE		DMAE	INIT	Reserved	
		rw	r	r	r	r	r	rw	rw	rw	rw	rw	w		

Bits 31:19 Reserved, forced by hardware to 0.

Bit 17 Reserved, forced by hardware to 0.

Bit 16 **LKEY:** Long key selection

This bit selects between short key (≤64 bytes) or long key (> 64 bytes) in HMAC mode

0: Short key (≤64 bytes)

1: Long key (> 64 bytes)

Note: This selection is only taken into account when the INIT bit is set and MODE = 1. Changing this bit during a computation has no effect.

Bits 15:14 Reserved, forced by hardware to 0.

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:** Number of words already pushed

This bitfield reflects the number of words in the message that have already been pushed into the IN FIFO.

NBW increments (+1) when a write access is performed to the HASH_DIN register while DINNE = 1.

It goes to 0000 when the INIT bit is written to 1 or when a digest calculation starts (DCAL written to 1 or DMA end of transfer).

- If the DMA is not used:

0000 and DINNE=0: no word has been pushed into the DIN buffer (the buffer is empty, both the HASH_DIN register and the IN FIFO are empty)

0000 and DINNE=1: 1 word has been pushed into the DIN buffer (The HASH_DIN register contains 1 word, the IN FIFO is empty)

0001: 2 words have been pushed into the DIN buffer (the HASH_DIN register and the IN FIFO contain 1 word each)

...

1111: 16 words have been pushed into the DIN buffer

- If the DMA is used, NBW is the exact number of words that have been pushed into the IN FIFO.

Bit 18 and bit 7 **ALGO[1:0]:** Algorithm selection

These bits selects the SHA-1, SHA-224, SHA256 or the MD5 algorithm:

00: SHA-1 algorithm selected

01: MD5 algorithm selected

10: SHA224 algorithm selected

11: SHA256 algorithm selected

Note: This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

Bit 6 **MODE:** Mode selection

This bit selects the HASH or HMAC mode for the selected algorithm:

0: Hash mode selected

1: HMAC mode selected. LKEY must be set if the key being used is longer than 64 bytes.

Note: This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

Bits 5:4 **DATATYPE:** Data type selection

Defines the format of the data entered into the HASH_DIN register:

00: 32-bit data. The data written into HASH_DIN are directly used by the HASH processing, without reordering.

01: 16-bit data, or half-word. The data written into HASH_DIN are considered as 2 half-words, and are swapped before being used by the HASH processing.

10: 8-bit data, or bytes. The data written into HASH_DIN are considered as 4 bytes, and are swapped before being used by the HASH processing.

11: bit data, or bit-string. The data written into HASH_DIN are considered as 32 bits (1st bit of the sting at position 0), and are swapped before being used by the HASH processing (1st bit of the string at position 31).

Bit 3 **DMAE:** DMA enable

0: DMA transfers disabled

1: DMA transfers enabled. A DMA request is sent as soon as the HASH core is ready to receive data.

Note: 1: This bit is cleared by hardware when the DMA asserts the DMA terminal count signal (while transferring the last data of the message). This bit is not cleared when the INIT bit is written to 1.

2: If this bit is written to 0 while a DMA transfer has already been requested to the DMA, DMAE is cleared but the current transfer is not aborted. Instead, the DMA interface remains internally enabled until the transfer is complete or INIT is written to 1.

Bit 2 **INIT:** Initialize message digest calculation

Writing this bit to 1 resets the hash processor core, so that the HASH is ready to compute the message digest of a new message.

Writing this bit to 0 has no effect.

Reading this bit always return 0.

Bits 1:0 Reserved, must be kept cleared.

25.4.3 HASH data input register (HASH_DIN)

Address offset: 0x04

Reset value: 0x0000 0000

HASH_DIN is the data input register. It is 32-bit wide. It is used to enter the message by blocks of 512 bits. When the HASH_DIN register is written to, the value presented on the AHB databus is ‘pushed’ into the HASH core and the register takes the new value presented on the AHB databus. The DATATYPE bits must previously have been configured in the HASH_CR register to get a correct message representation.

When a block of 16 words has been written to the HASH_DIN register, an intermediate digest calculation is launched:

- by writing new data into the HASH_DIN register (the first word of the next block) if the DMA is not used (intermediate digest calculation)
- automatically if the DMA is used

When the last block has been written to the HASH_DIN register, the final digest calculation (including padding) is launched:

- by writing the DCAL bit to 1 in the HASH_STR register (final digest calculation)
- automatically if the DMA is used and MDMAT bit is set to ‘0’.

When a digest calculation (intermediate or final) is in progress, any new write access to the HASH_DIN register is extended (by wait-state insertion on the AHB bus) until the HASH calculation completes.

When the HASH_DIN register is read, the last word written in this location is accessed (zero after reset).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAIN															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAIN															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATAIN:** Data input

Read = returns the current register content.
Write = the current register content is pushed into the IN FIFO, and the register takes the new value presented on the AHB databus.

25.4.4 HASH start register (HASH_STR)

Address offset: 0x08

Reset value: 0x0000 0000

The HASH_STR register has two functions:

- It is used to define the number of valid bits in the last word of the message entered in the hash processor (that is the number of valid least significant bits in the last data written into the HASH_DIN register)
- It is used to start the processing of the last block in the message by writing the DCAL bit to 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DCAL	Reserved			NBLW				
							w				rw	rw	rw	rw	rw

Bits 31:9 Reserved, forced by hardware to 0.

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.

Note

Bits 7:5 Reserved, forced by hardware to 0.

Bits 4:0 **NBLW:** Number of valid bits in the last word of the message in the bit string organization of hash processor

When these bits are written and DCAL is at '0', they take the value on the AHB databus:

0x00: All 32 bits of the last data written in the bit string organization of hash processor (after data swapping) are valid.

0x01: Only bit [31] of the last data written in the bit string organization of hash processor (after data swapping) are valid

0x02: Only bits [31:30] of the last data written in the bit string organization of hash processor (after data swapping) are valid

0x03: Only bits [31:29] of the last data written in the bit string organization of hash processor (after data swapping) are valid

...

0x1F: Only bits [0] of the last data written in the bit string organization of hash processor (after data swapping) are valid

When these bits are written and DCAL is at '1', the bitfield is not changed.

Reading them returns the last value written to NBLW.

Note: These bits must be configured before setting the DCAL bit, else they are not taken into account. Especially, it is not possible to configure NBLW and set DCAL at the same time.

25.4.5 HASH digest registers (HASH_HR0..4/5/6/7)

Address offset: 0x0C to 0x1C (STM32F415/417xx), plus 0x310 to 0x32C (STM32F43xxx)

Reset value: 0x0000 0000

These registers contain the message digest result named as:

1. H0, H1, H2, H3 and H4, respectively, in the SHA1 algorithm description
   Note that in this case, the HASH_H5 to HASH_H7 register is not used, and is read as zero.
2. A, B, C and D, respectively, in the MD5 algorithm description
   Note that in this case, the HASH_H4 to HASH_H7 register is not used, and is read as zero.
3. H0 to H6, respectively, in the SHA224 algorithm description,
   Note that in this case, the HASH_H7 register is not used, and is read as zero.
4. H0 to H7, respectively, in the SHA256 algorithm description,

If a read access to one of these registers occurs while the HASH core is calculating an intermediate digest or a final message digest (that is when the DCAL bit has been written to 1), then the read is stalled until the completion of the HASH calculation.

Note: H0, H1, H2, H3 and H4 mapping are duplicated in two region.

HASH_HR0

Address offset: 0x0C and 0x310

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H0															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H0															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

HASH_HR1

Address offset: 0x10 and 0x314

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H1															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H1															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

HASH_HR2

Address offset: 0x14 and 0x318

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H2															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H2															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

HASH_HR3

Address offset: 0x18 and 0x31C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H3															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H3															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

HASH_HR4

Address offset: 0x1C and 0x320

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

HASH_HR5

Address offset: 0x324

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H5															
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
H5															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

HASH_HR6

Address offset: 0x328

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H6															
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
H6															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

HASH_HR7

Address offset: 0x32C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H7															
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
H7															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Note: When starting a digest computation for a new bit stream (by writing the INIT bit to 1), these registers assume their reset values.

25.4.6 HASH interrupt enable register (HASH_IMR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														DCIE	DINIE
														rw	rw

Bits 31:2 Reserved, forced by hardware to 0.

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

25.4.7 HASH status register (HASH_SR)

Address offset: 0x24

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												BUSY	DMAS	DCIS	DINIS
												r	r	rc_w0	rc_w0

Bits 31:4 Reserved, forced by hardware to 0.

Bit 3 **BUSY:** Busy bit

0: No block is currently being processed
1: The hash core is processing a block of data

Bit 2 **DMAS:** DMA Status

This bit provides information on the DMA interface activity. It is set with DMAE and cleared when DMAE=0 and no DMA transfer is ongoing. No interrupt is associated with this bit.

0: DMA interface is disabled (DMAE=0) and no transfer is ongoing
1: DMA interface is enabled (DMAE=1) or a transfer is ongoing

Bit 1 **DCIS:** Digest calculation completion interrupt status

This bit is set by hardware when a digest becomes ready (the whole message has been processed). It is cleared by writing it to 0 or by writing the INIT bit to 1 in the HASH_CR register.

0: No digest available in the HASH_Hx registers
1: Digest calculation complete, a digest is available in the HASH_Hx registers. An interrupt is generated if the DCIE bit is set in the HASH_IMR register.

Bit 0 **DINIS:** Data input interrupt status

This bit is set by hardware when the input buffer is ready to get a new block (16 locations are free). It is cleared by writing it to 0 or by writing the HASH_DIN register.

0: Less than 16 locations are free in the input buffer
1: A new block can be entered into the input buffer. An interrupt is generated if the DINIE bit is set in the HASH_IMR register.

25.4.8 HASH context swap registers (HASH_CSRx)

Address offset: 0x0F8 to 0x1C0

- For HASH_CSR0 register: Reset value is 0x0000 0002.
- For others registers: Reset value is 0x0000 0000 , except for STM32F43xxx devices where the HASH_CSR2 register reset value is 0x2000 0000

Additional registers are available from 0x1C1 to 0x1CC on STM32F43xxx:

- Reset value: 0x0000 0000.

These registers contain the complete internal register states of the hash processor, and are useful when a context swap has to be done because a high-priority task has to use the hash processor while it is already in use by another task.

When such an event occurs, the HASH_CSRx registers have to be read and the read values have to be saved somewhere in the system memory space. Then the hash processor can be used by the preemptive task, and when hash computation is finished, the saved context can be read from memory and written back into these HASH_CSRx registers.

HASH_CSRx

Address offset: 0x0F8 to 0x1C0 on STM32F415/417xx

Address offset: 0x0F8 to 0x1CC on STM32F43xxx

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

25.4.9 HASH register map

Table 9 gives the summary HASH register map and reset values.

Table 117. HASH register map and reset values on STM32F415/417xx

Offset	Register name reset value	Register size																													
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	HASH_CR	Reserved															LKEY	Reserved			DINNE	NBW				ALGO[0]	MODE	DATATYPE		DMAE	INIT	Reserved	
	Reset value																0				0	0	0	0	0	0	0	0	0	0	0		
0x04	HASH_DIN	DATAIN																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	HASH_STR	Reserved																							DCAL	Reserved				NBLW			
	Reset value																								0					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	Reserved																													DCIE	DINIE	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	HASH_SR	Reserved																												BUSY	DMAS	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
	...																																
	...																																
0x1C0	HASH_CSR50	CSR50																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reserved																																
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	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	0

Table 117. HASH register map and reset values on STM32F415/417xx (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
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	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	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	0

Table 118. HASH register map and reset values on STM32F43xxx

Offset	Register name reset value	Register size																													
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	HASH_CR	Reserved													ALGO[1]	Reserved	LKEY	Reserved		MDMAT	DINNE	NBW				ALGO[0]	MODE	DATATYPE		DMAE	INIT	Reserved	
	Reset value														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	Reserved																							DCAL	Reserved				NBLW			
	Reset value																								0					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	Reserved																													DCIE	DINIE	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	HASH_SR	Reserved																												BUSY	DMAS	DCIS	DINIS
	Reset value																													0	0	0	1

Table 118. HASH register map and reset values on STM32F43xxx (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
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
...																																	
...																																	
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	0
Reserved																																	
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	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	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	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	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	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	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	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	0

26 Real-time clock (RTC)

This section applies to the whole STM32F4xx family, unless otherwise specified.

26.1 Introduction

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar, two programmable alarm interrupts, and a periodic programmable wakeup flag with interrupt capability. The RTC also includes an automatic wakeup unit to manage low-power modes.

Two 32-bit registers contain the seconds, minutes, hours (12- or 24-hour format), day (day of week), date (day of month), month, and year, expressed in binary coded decimal format (BCD). The sub-seconds value is also available in binary format.

Compensations for 28-, 29- (leap year), 30-, and 31-day months are performed automatically. Daylight saving time compensation can also be performed.

Additional 32-bit registers contain the programmable alarm subseconds, seconds, minutes, hours, day, and date.

A digital calibration feature is available to compensate for any deviation in crystal oscillator accuracy.

After 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).

26.2 RTC main features

The RTC unit main features are the following (see *Figure 237: RTC block diagram*):

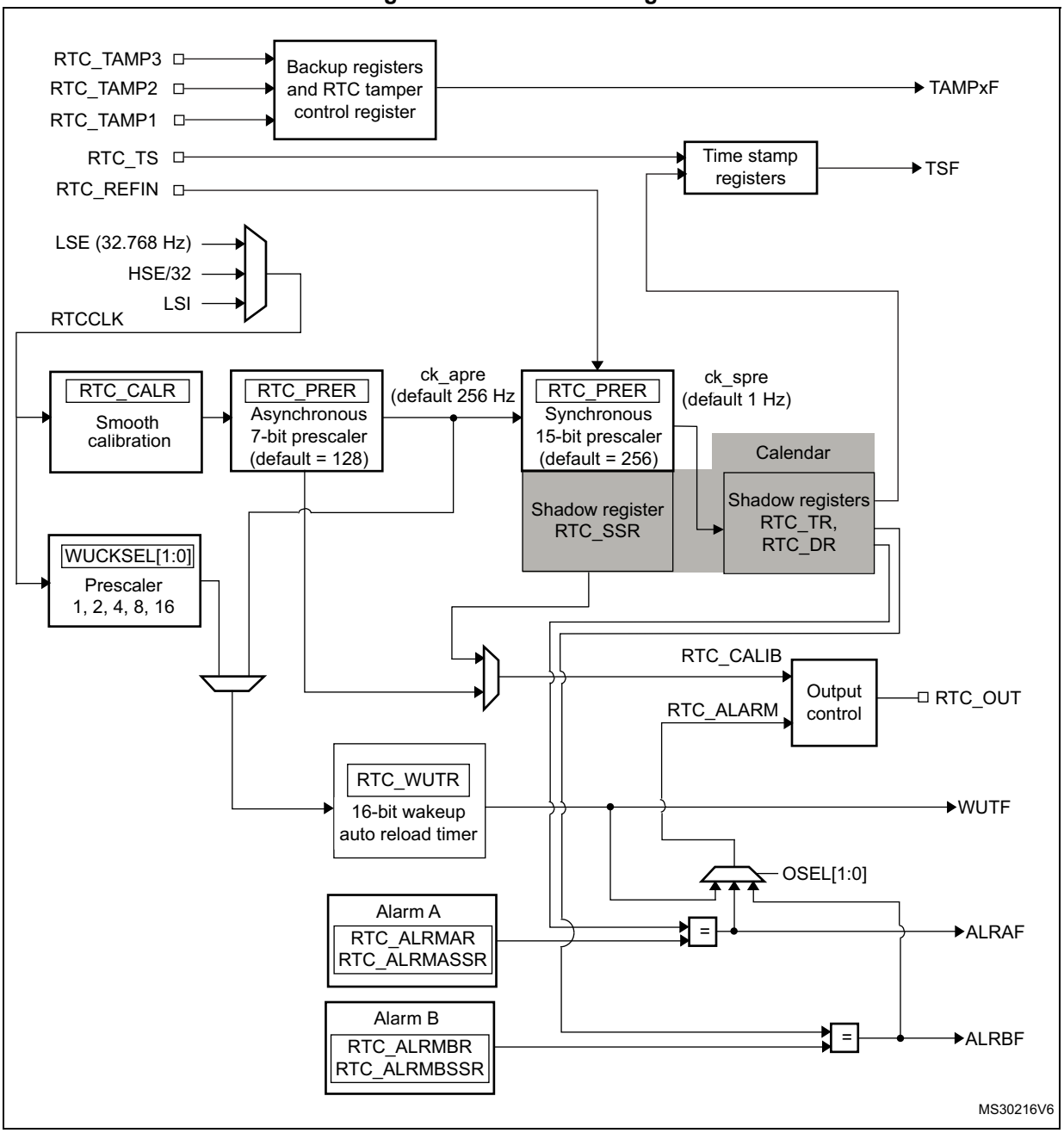
- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), day (day of week), date (day of month), month, and year.
- Daylight saving compensation programmable by software.
- Two programmable alarms with interrupt function. The alarms can be triggered by any combination of the calendar fields.
- Automatic wakeup unit generating a periodic flag that triggers an automatic wakeup interrupt.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Accurate synchronization with an external clock using the subsecond shift feature.
- Maskable interrupts/events:
  - Alarm A
  - Alarm B
  - Wakeup interrupt
  - Timestamp
  - Tamper detection
- Digital calibration circuit (periodic counter correction)
  - 5 ppm accuracy
  - 0.95 ppm accuracy, obtained in a calibration window of several seconds
- Timestamp function for event saving (1 event)
- Tamper detection:
  - 2 tamper events with configurable filter and internal pull-up.
- 20 backup registers (80 bytes). The backup registers are reset when a tamper detection event occurs.
- Alternate function output (RTC_OUT) which selects one of the following two outputs:
  - RTC_CALIB: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz).

    This output is enabled by setting the COE bit in the RTC_CR register. It is routed to the device RTC_AF1 function.
  - RTC_ALARM (Alarm A, Alarm B or wakeup).

    This output is selected by configuring the OSEL[1:0] bits in the RTC_CR register. It is routed to the device RTC_AF1 function.
- RTC alternate function inputs:
  - RTC_TS: timestamp event detection. It is routed to the device RTC_AF1 and RTC_AF2 functions.
  - RTC_TAMP1: TAMPER1 event detection. It is routed to the device RTC_AF1 and RTC_AF2 functions.
  - RTC_TAMP2: TAMPER2 event detection.
  - RTC_REFIN: reference clock input (usually the mains, 50 or 60 Hz).

Refer to Section 8.3.15: Selection of RTC_AF1 and RTC_AF2 alternate functions.

Figure 237. RTC block diagram



1. On STM32F4xx devices, the RTC_AF1 and RTC_AF2 alternate functions are connected to PC13 and PI8, respectively.

26.3 RTC functional description

26.3.1 Clock and prescalers

The RTC clock source (RTCCLK) is selected through the clock controller among the LSE clock, the LSI oscillator clock, and the HSE clock. For more information on the RTC clock source configuration, refer to *Section 7: Reset and clock control for STM32F405xx/07xx and STM32F415xx/17xx(RCC)* and *Section 6: Reset and clock control for STM32F42xxx and STM32F43xxx (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 237: RTC block diagram*):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is 2^{22}.

This corresponds to a maximum input frequency of around 4 MHz.

f_{ck_apre} is given by the following formula:

$$f_{CK\_APRE} = \frac{f_{RTCCLK}}{PREDIV\_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subseconds downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

f_{ck_spre} is given by the following formula:

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(PREDIV\_S + 1) \times (PREDIV\_A + 1)}$$

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wakeup auto-reload timer. To obtain short timeout periods, the 16-bit wakeup auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 26.3.4: Periodic auto-wakeup* for details).

26.3.2 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK1 (APB1 clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every two RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ISR register is set (see *Section 26.6.4*). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 2 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers.t is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD=0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

26.3.3 Programmable alarms

The RTC unit provides two programmable alarms, Alarm A and Alarm B.

The programmable alarm functions are enabled through the ALRAIE and ALRBIE bits in the RTC_CR register. The ALRAF and ALRBF flags are set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR/RTC_ALRMAR and RTC_ALRMBSSR/RTC_ALRMBR, respectively. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR and RTC_ALRMBR registers, and through the MASKSSx bits of the RTC_ALRMASSR and RTC_ALRMBSSR registers. The alarm interrupts are enabled through the ALRAIE and ALRBIE bits in the RTC_CR register.

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 polarity can be configured through bit POL in the RTC_CR register.

Caution: If the seconds field is selected (MSK0 bit reset in RTC_ALRMAR or RTC_ALRMBR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

26.3.4 Periodic auto-wakeup

The periodic wakeup flag is generated by a 16-bit programmable auto-reload down-counter. The wakeup timer range can be extended to 17 bits.

The wakeup function is enabled through the WUTE bit in the RTC_CR register.

The wakeup timer clock input can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSE(32.768kHz), this allows configuring the wakeup interrupt period from 122 µs to 32 s, with a resolution down to 61µs.
- ck_spre (usually 1 Hz internal clock)

  When ck_spre frequency is 1Hz, this allows achieving 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^{16} is added to the 16-bit counter current value.When the initialization sequence is complete (see *Programming the wakeup timer on page 805*), the timer starts counting down.When the wakeup function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_ISR register, and the wakeup counter is automatically reloaded with its reload value (RTC_WUTR register value).

The WUTF flag must then be cleared by software.

When the periodic wakeup interrupt is enabled by setting the WUTIE bit in the RTC_CR2 register, it can exit the device from low-power modes.

The periodic wakeup flag can be routed to the RTC_ALARM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. RTC_ALARM 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.

26.3.5 RTC initialization and configuration

RTC register access

The RTC registers are 32-bit registers. The APB interface introduces 2 wait-states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD=0.

RTC register write protection

After system reset, the RTC registers are protected against parasitic write access with the DBP bit of the PWR power control register (PWR_CR). The DBP bit must be set 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_ISR[13:8], RTC_TAFCR, and RTC_BKPxR.

1. Write ‘0xCA’ into the RTC_WPR register.
2. Write ‘0x53’ into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ISR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ISR register. The initialization phase mode is entered when INITF is set to 1. It takes from 1 to 2 RTCCLK clock cycles (due to clock synchronization).
3. To generate a 1 Hz clock for the calendar counter, program first the synchronous prescaler factor in RTC_PRER register, and then program the asynchronous prescaler factor. Even if only one of the two fields needs to be changed, 2 separate write accesses must be performed to the 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 alarm (Alarm A or Alarm B):

1. Clear ALRAE or ALRBIE in RTC_CR to disable Alarm A or Alarm B.
2. Poll ALRAWF or ALRBWF in RTC_ISR until it is set to make sure the access to alarm registers is allowed. This takes 1 to 2 RTCCLK clock cycles (due to clock synchronization).
3. Program the Alarm A or Alarm B registers (RTC_ALRMASSR/RTC_ALRMAR or RTC_ALRMBSSR/RTC_ALRMBR).
4. Set ALRAE or ALRBIE in the RTC_CR register to enable Alarm A or Alarm B again.

Note: Each change of the RTC_CR register is taken into account after 1 to 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 1 to 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 clocks cycles after WUTE is cleared, due to clock synchronization.

26.3.6 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB1 clock frequency (f_{PCLK1}) must be equal to or greater than seven times the f_{RTCCLK} RTC clock frequency. This ensures a secure behavior of the synchronization mechanism.

If the APB1 clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB1 clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ISR register each time the calendar registers are copied into the RTC_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.

Note: 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 804): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to Section 26.3.8: RTC synchronization): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (STOP or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD=1, instructions which read the calendar registers require one extra APB cycle to complete.

26.3.7 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 resetted 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 registers (RTC_CALIBR or RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the RTC tamper and alternate function configuration register (RTC_TAFCR), the RTC backup registers (RTC_BKPxR), the wakeup timer register (RTC_WUTR), the Alarm A and Alarm B registers (RTC_ALRMASSR/RTC_ALRMAR and RTC_ALRMBSSR/RTC_ALRMBR).

In addition, when the RTC is clocked by the LSE, it goes on running under system reset if the reset source is different from a backup domain reset one. Refer to section *RTC clock* of the *Reset and clock controller* for details about the list of the RTC clock sources that are 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.

26.3.8 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

RTC_SSR contains the value of the synchronous prescaler’s counter. This allows one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]. The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler's output at 1 Hz. In this way, the frequency of the asynchronous prescaler's output increases, which may increase the RTC dynamic consumption.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock by up to a second with a resolution of 1 / (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.

26.3.9 RTC reference clock detection

The RTC calendar update can be synchronized to a reference clock RTC_REFIN, usually the mains (50 or 60 Hz). The RTC_REFIN reference clock should have a higher precision 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 reference clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

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 reference clock detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREDIV_S = 0x00FF

Note: The reference clock detection is not available in Standby mode.

Caution: The reference clock detection feature cannot be used in conjunction with the coarse digital calibration: RTC_CALIBR must be kept at 0x0000 0000 when REFCKON=1.

26.3.10 RTC coarse digital calibration

Two digital calibration methods are available: coarse and smooth calibration. To perform coarse calibration refer to *Section 26.6.7: RTC calibration register (RTC_CALIBR)*.

The two calibration methods are not intended to be used together, the application must select one of the two methods. Coarse calibration is provided for compatibly reasons. To perform smooth calibration refer to *Section 26.3.11: RTC smooth digital calibration* and the *Section 26.6.16: RTC calibration register (RTC_CALR)*

The coarse digital calibration can be used to compensate crystal inaccuracy by adding (positive calibration) or masking (negative calibration) clock cycles at the output of the asynchronous prescaler (ck_apre).

Positive and negative calibration are selected by setting the DCS bit in RTC_CALIBR register to ‘0’ and ‘1’, respectively.

When positive calibration is enabled (DCS = ‘0’), 2 ck_apre cycles are added every minute (around 15360 ck_apre cycles) for 2xDC minutes. This causes the calendar to be updated sooner, thereby adjusting the effective RTC frequency to be a bit higher.

When negative calibration is enabled (DCS = ‘1’), 1 ck_apre cycle is removed every minute (around 15360 ck_apre cycles) for 2xDC minutes. This causes the calendar to be updated later, thereby adjusting the effective RTC frequency to be a bit lower.

DC is configured through bits DC[4:0] of RTC_CALIBR register. This number ranges from 0 to 31 corresponding to a time interval (2xDC) ranging from 0 to 62.

The coarse digital calibration can be configured only in initialization mode, and starts when the INIT bit is cleared. The full calibration cycle lasts 64 minutes. The first 2xDC minutes of the 64 -minute cycle are modified as just described.

Negative calibration can be performed with a resolution of about 2 ppm while positive calibration can be performed with a resolution of about 4 ppm. The maximum calibration ranges from -63 ppm to 126 ppm.

The calibration can be performed either on the LSE or on the HSE clock.

Caution: Digital calibration may not work correctly if PREDIV_A < 6.

Case of RTCCLK=32.768 kHz and PREDIV_A+1=128

The following description assumes that ck_apre frequency is 256 Hz obtained with an LSE clock nominal frequency of 32.768 kHz, and PREDIV_A set to 127 (default value).

The ck_spre clock frequency is only modified during the first 2xDC minutes of the 64-minute cycle. For example, when DC equals 1, only the first 2 minutes are modified. This means that the first 2xDC minutes of each 64-minute cycle have, once per minute, one second

either shortened by 256 or lengthened by 128 RTCCLK cycles, given that each ck_apre cycle represents 128 RTCCLK cycles (with PREDIV_A+1=128).

Therefore each calibration step has the effect of adding 512 or subtracting 256 oscillator cycles for every 125829120 RTCCLK cycles (64min x 60 s/min x 32768 cycles/s). This is equivalent to +4.069 ppm or -2.035 ppm per calibration step. As a result, the calibration resolution is +10.5 or -5.27 seconds per month, and the total calibration ranges from +5.45 to -2.72 minutes per month.

In order to measure the clock deviation, a 512 Hz clock is output for calibration.Refer to *Section 26.3.14: Calibration clock output*.

26.3.11 RTC smooth digital calibration

RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual RTCCLK pulses). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

The smooth digital calibration is performed during a cycle of about 2^{20} RTCCLK pulses, or 32 seconds when the input frequency is 32768 Hz. 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_CALRx) specifies the number of RTCCLK pulses to be masked during the 32-second cycle. Setting the bit CALM[0] to '1' causes exactly one pulse to be masked during the 32-second cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1]=1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2]=1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8]=1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM allows the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to '1' effectively inserts an extra RTCCLK pulse every 2^{11} RTCCLK cycles, which means that 512 clocks are added during every 32-second cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 RTCCLK cycles can be added during the 32-second cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Calibration when PREDIV_A<3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are

set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

To perform a calibration with PREDIV_A less than 3, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 RTCCLK clock cycles, which is equivalent to adding 256 clock cycles every 32 seconds. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each 32-second cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the

calibrated input clock is as follows:

$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Verifying the RTC calibration

RTC precision is performed by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

  Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).
- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

  In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.
- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8- second calibration cycle period.

  In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are 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.

26.3.12 Timestamp function

Timestamp is enabled by setting the TSE bit of RTC_CR register to 1.

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the pin to which the TIMESTAMP alternate function is mapped. When a timestamp event occurs, the timestamp flag bit (TSF) in RTC_ISR register is set.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

Note: TSF is set 2 ck_apre cycles after the timestamp event occurs due to synchronization process.

There is no delay in the setting of TSOVF. This means that if two timestamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

Caution: If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write ‘0’ into TSF bit unless it has already read it to ‘1’.

Optionally, a tamper event can cause a timestamp to be recorded. See the description of the TAMPTS control bit in *Section 26.6.17: RTC tamper and alternate function configuration register (RTC_TAFCR)*. If the timestamp event is on the same pin as a tamper event configured in filtered mode (TAMPFLT set to a non-zero value), the timestamp on tamper detection event mode must be selected by setting TAMPTS='1' in RTC_TAFCR register.

TIMESTAMP alternate function

The TIMESTAMP alternate function (RTC_TS) can be mapped either to RTC_AF1 or to RTC_AF2 depending on the value of the TSINSEL bit in the RTC_TAFCR register (see *Section 26.6.17: RTC tamper and alternate function configuration register (RTC_TAFCR)*). Mapping the timestamp event on RTC_AF2 is not allowed if RTC_AF1 is used as TAMPER in filtered mode (TAMPFLT set to a non-zero value).

26.3.13 Tamper detection

Two tamper detection inputs are available. They can be configured either for edge detection, or for level detection with filtering.

RTC backup registers

The backup registers (RTC_BKPxR) are twenty 32-bit registers for storing 80 bytes of user application data. They are implemented in the backup domain that remains powered-on by

V_{BAT} when the V_{DD} power is switched off. They are not reset by system reset or when the device wakes up from Standby mode. They are reset by a backup domain reset

The backup registers are reset when a tamper detection event occurs (see *Section 26.6.20: RTC backup registers (RTC_BKPxR)* and *Tamper detection initialization on page 812*.

Tamper detection initialization

Each tamper detection input is associated with the TAMP1F/TAMP2F flags in the RTC_ISR2 register. Each input can be enabled by setting the corresponding TAMP1E/TAMP2E bits to 1 in the RTC_TAFCR register.

A tamper detection event resets all backup registers (RTC_BKPxR).

By setting the TAMPIE bit in the RTC_TAFCR register, an interrupt is generated when a tamper detection event occurs.

Timestamp on tamper event

With TAMPTS set to ‘1 , any tamper event causes a timestamp to occur. In this case, either the TSF bit or the TSOVF bit are set in RTC_ISR, in the same manner as if a normal timestamp event occurs. The affected tamper flag register (TAMP1F, TAMP2F) is set at the same time that TSF or TSOVF is set.

Edge detection on tamper inputs

If the TAMPFLT bits are “00”, the TAMPER pins generate tamper detection events (RTC_TAMP[2:1]) when either a rising edge is observed or an falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMPER inputs are deactivated when edge detection is selected.

Caution: To avoid losing tamper detection events, the signal used for edge detection is logically ANDed with TAMPxE in order to detect a tamper detection event in case it occurs before the TAMPERx pin is enabled.

- When TAMPxTRG = 0: if the TAMPERx alternate function is already high before tamper detection is enabled (TAMPxE bit set to 1), a tamper event is detected as soon as TAMPERx is enabled, even if there was no rising edge on TAMPERx after TAMPxE was set.
- When TAMPxTRG = 1: if the TAMPERx alternate function is already low before tamper detection is enabled, a tamper event is detected as soon as TAMPERx is enabled (even if there was no falling edge on TAMPERx after TAMPxE was set.

After a tamper event has been detected and cleared, the TAMPERx alternate function should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (RTC_BKPxR). This prevents the application from writing to the backup registers while the TAMPERx value still indicates a tamper detection. This is equivalent to a level detection on the TAMPERx alternate function.

Note: Tamper detection is still active when V_{DD} power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the TAMPER alternate function is mapped should be externally tied to the correct level.

Level detection with filtering on tamper inputs

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive

samples are observed at the level designated by the TAMPxTRG bits (TAMP1TRG/TAMP2TRG).

The TAMPER inputs are pre-charged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1,The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the tamper inputs.

The 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.

TAMPER alternate function detection

The TAMPER1 alternate function (RTC_TAMP1) can be mapped either to RTC_AF1(PC13) or RTC_AF2 (PI8) depending on the value of TAMP1INSEL bit in RTC_TAFCR register (see *Section 26.6.17: RTC tamper and alternate function configuration register (RTC_TAFCR)*). TAMPE bit must be cleared when TAMP1INSEL is modified to avoid unwanted setting of TAMPF.

The TAMPER 2 alternate function corresponds to RTC_TAMP2 pin.

26.3.14 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the RTC_CALIB device output. If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the RTC_CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz.

The RTC_CALIB output is not impacted by the calibration value programmed in RTC_CALIBR register. The RTC_CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

If COSEL is set and (PREDIV_S+1) is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the RTC_CALIB frequency is fRTCCLK/(256 * (PREDIV_A+1)). This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = 0x7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz. The 1 Hz output is affected when a shift operation is on going and may toggle during the shift operation (SHPF=1).

Calibration alternate function output

When the COE bit in the RTC_CR register is set to 1, the calibration alternate function (RTC_CALIB) is enabled on RTC_AF1.

Note: When RTC_CALIB *or* RTC_ALARM *is selected,* RTC_AF1 *is automatically configured in output alternate function.*

26.3.15 Alarm output

Three functions can be selected on Alarm output: ALRAF, ALRBF and WUTF. These functions reflect the contents of the corresponding flags in the RTC_ISR register.

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm alternate function output (RTC_ALARM) in RTC_AF1, and to select the function which is output on RTC_ALARM.

The polarity of the output is determined by the POL control bit in RTC_CR so that the opposite of the selected flag bit is output when POL is set to 1.

Alarm alternate function output

RTC_ALARM can be configured in output open drain or output push-pull using the control bit ALARMOUTTYPE in the RTC_TAFCR register.

Note: *Once* RTC_ALARM *is enabled, it has priority over* RTC_CALIB *(*COE *bit is don't care on* RTC_AF1*).*

When RTC_CALIB *or* RTC_ALARM *is selected,* RTC_AF1 *is automatically configured in output alternate function.*

26.4 RTC and low-power modes

Table 119. Effect of low-power modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop	The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC time stamp event, and RTC Wakeup cause the device to exit the Stop mode.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC time stamp event, and RTC Wakeup cause the device to exit the Standby mode.

26.5 RTC interrupts

All RTC interrupts are connected to the EXTI controller.

To enable the RTC Alarm interrupt, the following sequence is required:

1. Configure and enable the EXTI Line 17 in interrupt mode and select the rising edge sensitivity.
2. Configure and enable the RTC_Alarm IRQ channel in the NVIC.
3. Configure the RTC to generate RTC alarms (Alarm A or Alarm B).

To enable the RTC Wakeup interrupt, the following sequence is required:

1. Configure and enable the EXTI Line 22 in interrupt mode and select the rising edge sensitivity.
2. Configure and enable the RTC_WKUP IRQ channel in the NVIC.
3. Configure the RTC to generate the RTC wakeup timer event.

To enable the RTC Tamper interrupt, the following sequence is required:

1. Configure and enable the EXTI Line 21 in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the TAMP_STAMP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC tamper event.

To enable the RTC TimeStamp interrupt, the following sequence is required:

1. Configure and enable the EXTI Line 21 in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the TAMP_STAMP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC timestamp event.

Table 120. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Exit the Sleep mode	Exit the Stop mode	Exit the Standby mode
Alarm A	ALRAF	ALRAIE	yes	yes(1)	yes(1)
Alarm B	ALRBF	ALRBIE	yes	yes(1)	yes(1)
Wakeup	WUTF	WUTIE	yes	yes(1)	yes(1)
TimeStamp	TSF	TSIE	yes	yes(1)	yes(1)
Tamper1 detection	TAMP1F	TAMPIE	yes	yes(1)	yes(1)
Tamper2 detection(2)	TAMP2F	TAMPIE	yes	yes(1)	yes(1)

1. Wakeup from STOP and Standby modes is possible only when the RTC clock source is LSE or LSI.
2. If RTC_TAMPER2 pin is present. Refer to device datasheet pinout.

26.6 RTC registers

Refer to *Section 1.1: List of abbreviations for registers for registers* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

26.6.1 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 804* and *Reading the calendar on page 805*.

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
Reserved									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 31-24 Reserved

Bit 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

Bit 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

Note: This register is write protected. The write access procedure is described in RTC register write protection on page 803.

26.6.2 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 804* and *Reading the calendar on page 805*.

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
Reserved								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]				Reserved		DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31-24 Reserved

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units
000: forbidden
001: Monday
...
111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

Note: This register is write protected. The write access procedure is described in RTC register write protection on page 803.

26.6.3 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
Reserved								COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
								rw	rw	rw	rw	rw	rw	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	DCE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **COE**: Calibration output enable

This bit enables the RTC_CALIB output

0: Calibration output disabled

1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection

These bits are used to select the flag to be routed to RTC_ALARM output

00: Output disabled

01: Alarm A output enabled

10:Alarm B output enabled

11: Wakeup output enabled

Bit 20 **POL**: Output polarity

This bit is used to configure the polarity of RTC_ALARM output

0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0])

1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]).

Bit 19 **COSEL**: Calibration output selection

When COE=1, this bit selects which signal is output on RTC_CALIB.

0: Calibration output is 512 Hz

1: Calibration output is 1 Hz

These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A=127 and PREDIV_S=255). Refer to *Section 26.3.14: Calibration clock output*

Bit 18 **BKP**: Backup

This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: Subtract 1 hour (winter time change)

When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.

Setting this bit has no effect when current hour is 0.

0: No effect

1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)
When this bit is set, 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**: Timestamp interrupt enable
0: Timestamp Interrupt disable
1: Timestamp Interrupt enable

Bit 14 **WUTIE**: Wakeup timer interrupt enable
0: Wakeup timer interrupt disabled
1: Wakeup timer interrupt enabled

Bit 13 **ALRBIE**: *Alarm B interrupt enable*
0: Alarm B Interrupt disable
1: Alarm B Interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable
0: Alarm A interrupt disabled
1: Alarm A interrupt enabled

Bit 11 **TSE**: Time stamp enable
0: Time stamp disable
1: Time stamp enable

Bit 10 **WUTE**: Wakeup timer enable
0: Wakeup timer disabled
1: Wakeup timer enabled
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 **DCE:** Coarse digital calibration enable
0: Digital calibration disabled
1: Digital calibration enabled
PREDIV_A must be 6 or greater

Bit 6 **FMT**: Hour format
0: 24 hour/day format
1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers
0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.
1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.
Note: If the frequency of the APB1 clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to '1'.

Bit 4 **REFCKON**: Reference clock detection enable (50 or 60 Hz)
0: Reference clock detection disabled
1: Reference clock detection enabled
Note: PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Timestamp event active edge
0: TIMESTAMP rising edge generates a timestamp event
1: TIMESTAMP falling edge generates a timestamp event
TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting

Bits 2:0 **WUCKSEL[2:0]**: Wakeup clock selection
000: RTC/16 clock is selected
001: RTC/8 clock is selected
010: RTC/4 clock is selected
011: RTC/2 clock is selected
10x: ck_spre (usually 1 Hz) clock is selected
11x: ck_spre (usually 1 Hz) clock is selected and 2^{16} is added to the WUT counter value (see note below)

Note: WUT = Wakeup unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 7, 6 and 4 of this register can be written in initialization mode only (RTC_ISR/INITF = 1).

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.

To avoid spuriously setting of TSF, TSE must be reset when TSEDGE is changed.

This register is write protected. The write access procedure is described in RTC register write protection on page 803.

26.6.4 RTC initialization and status register (RTC_ISR)

Address offset: 0x0C

Backup domain reset value: 0x0000 0007

System reset value: Not affected except INIT, INITF and RSF which are cleared to 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRBWF	ALRAWF
	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	r	rc_w0	r	r	r	r	r

Bits 31:17 Reserved

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 *Section : Re-calibration on-the-fly*.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TAMP2F**: TAMPER2 detection flag

This flag is set by hardware when a tamper detection event is detected on tamper input 2.
It is cleared by software writing 0.

Bit 13 **TAMP1F**: Tamper detection flag

This flag is set by hardware when a tamper detection event is detected.
It is cleared by software writing 0.

Bit 12 **TSOVF**: Timestamp overflow flag

This flag is set by hardware when a timestamp event occurs while TSF is already set.
This flag is cleared by software by writing 0. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 11 **TSF**: Timestamp flag

This flag is set by hardware when a timestamp event occurs.
This flag is cleared by software by writing 0.

Bit 10 **WUTF**: Wakeup timer flag

This flag is set by hardware when the wakeup auto-reload counter reaches 0.
This flag is cleared by software by writing 0.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 9 **ALRBF**: Alarm B flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the Alarm B register (RTC_ALRMBR).
This flag is cleared by software by writing 0.

Bit 8 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the Alarm A register (RTC_ALRMAR).
This flag is cleared by software by writing 0.

Bit 7 **INIT**: Initialization mode

0: Free running mode
1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER). Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.
0: Calendar registers update is not allowed
1: Calendar registers update is allowed.

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSRx, RTC_TRx and RTC_DRx). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF=1), or when in bypass shadow register mode (BYPSHAD=1). This bit can also be cleared by software.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (backup domain reset value state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

0: No shift operation is pending

1: A shift operation is pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR. It is cleared by hardware when the corresponding shift operation has been executed. Writing to SHPF has no effect.

Bit 2 **WUTWF**: Wakeup timer write flag

This bit is set by hardware 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 ALRBIE 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 ALRAF, ALRBF, WUTF and TSF bits are cleared 2 APB clock cycles after programming them to 0.

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 803.*

26.6.5 RTC prescaler register (RTC_PRER)

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
Reserved									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 31:24 Reserved

Bit 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)

Note: 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 804

This register is write protected. The write access procedure is described in RTC register write protection on page 803.

26.6.6 RTC wakeup timer register (RTC_WUTR)

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved

Bits 15:0 **WUT[15:0]**: Wakeup auto-reload value bits

When the wakeup timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register

When WUCKSEL[2] = 1, the wakeup timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

Note: The first assertion of WUTF occurs (WUT+1) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] =011 (RTCCLK/2) is forbidden.

Note: 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 803.

26.6.7 RTC calibration register (RTC_CALIBR)

Address offset: 0x18

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
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DCS	Reserved		DC[4:0]				
								rw			rw	rw	rw	rw	rw

Bits 31:8 Reserved

Bit 7 **DCS**: Digital calibration sign

0: Positive calibration: calendar update frequency is increased

1: Negative calibration: calendar update frequency is decreased

Bits 6:5 Reserved, must be kept at reset value.

Bits 4:0 **DC[4:0]**: Digital calibration

DCS = 0 (positive calibration)

00000: + 0 ppm

00001: + 4 ppm (rounded value)

00010: + 8 ppm (rounded value)

..

11111: + 126 ppm (rounded value)

DCS = 1 (negative calibration)

00000: –0 ppm

00001: –2 ppm (rounded value)

00010: –4 ppm (rounded value)

..

11111: –63 ppm (rounded value)

Refer to *Case of RTCCLK=32.768 kHz and PREDIV_A+1=128 on page 808* for the exact step value.

Note: This register can be written in initialization mode only (RTC_ISR/INITF = '1').

This register is write protected. The write access procedure is described in RTC register write protection on page 803.

26.6.8 RTC alarm A register (RTC_ALRMAR)

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.

Note: 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 803.

26.6.9 RTC alarm B register (RTC_ALRMBR)

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

Note: 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 803.

26.6.10 RTC write protection register (RTC_WPR)

Address offset: 0x24

Backup domain reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								KEY							
								w	w	w	w	w	w	w	w

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.

26.6.11 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
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved

Bits 15:0 **SS**: Sub second value

SS[15:0] is the value in the synchronous prescaler's counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

Note: SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

26.6.12 RTC shift control register (RTC_SHIFTR)

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	Reserved														
w	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.	SUBFS[14:0]														
r	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

Bits 14:0 **SUBFS**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC_ISR).

The value which is written to SUBFS is added to the synchronous prescaler's counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))) .

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF=1 to be sure that the shadow registers have been updated with the shifted time.

Refer to *Section 26.3.8: RTC synchronization*.

Note: This register is write protected. The write access procedure is described in RTC register write protection on page 803

26.6.13 RTC time stamp time register (RTC_TSTR)

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
Reserved									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.

Note: The content of this register is valid only when TSF is set to 1 in RTC_ISR. It is cleared when TSF bit is reset.

26.6.14 RTC time stamp date register (RTC_TSDR)

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[1:0]			MT	MU[3:0]				Reserved		DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **WDU[1:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bit 3:0 **DU[3:0]**: Date units in BCD format

Note: The content of this register is valid only when TSF is set to 1 in RTC_ISR. It is cleared when TSF bit is reset.

26.6.15 RTC timestamp sub second register (RTC_TSSSR)

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
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved

Bits 15:0 **SS**: Sub second value

SS[15:0] is the value of the synchronous prescaler's counter when the timestamp event occurred.

Note: The content of this register is valid only when RTC_ISR/TSF is set. It is cleared when the RTC_ISR/TSF bit is reset.

26.6.16 RTC calibration register (RTC_CALR)

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
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW16	Reserved				CALM[8:0]								
rw	rw	rw	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved

Bit 15 CALP: Increase frequency of RTC by 488.5 ppm
0: No RTCCLK pulses are added.
1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm).
This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows: (512 * CALP) - CALM.
Refer to *Section 26.3.11: RTC smooth digital calibration*.

Bit 14 **CALW8:** Use an 8-second calibration cycle period
When CALW8 is set to '1', the 8-second calibration cycle period is selected.
CALM[1:0] are stuck at "00" when CALW8='1'.
Refer to *Section 26.3.11: RTC smooth digital calibration*.

Bit 13 **CALW16:** Use a 16-second calibration cycle period
When CALW16 is set to '1', the 16-second calibration cycle period is selected. This bit must not be set to '1' if CALW8=1.
Note: CALM[0] is stuck at '0' when CALW16='1'.
Refer to *Section 26.3.11: RTC smooth digital calibration*.

Bits 12:9 Reserved

Bits 8:0 **CALM[8:0]**: Calibration minus
The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.
To increase the frequency of the calendar, this feature should be used in conjunction with CALP.
See *Section 26.3.11: RTC smooth digital calibration on page 809*.

Note: This register is write protected. The write access procedure is described in RTC register write protection on page 803

26.6.17 RTC tamper and alternate function configuration register (RTC_TAFCR)

Address offset: 0x40

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ALARMOUTTYPE	TSINSEL	TAMP1INSEL
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMP-PUDIS	TAMP-PRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMPTS	Reserved		TAMP2-TRG	TAMP2E	TAMPIE	TAMP1TRG	TAMP1E
rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw

Bits 31:19 Reserved. Always read as 0.

Bit 18 **ALARMOUTTYPE**: RTC_ALARM output type
0: RTC_ALARM is an open-drain output
1: RTC_ALARM is a push-pull output

Bit 17 **TSINSEL**: TIMESTAMP mapping
0: RTC_AF1 used as TIMESTAMP
1: RTC_AF2 used as TIMESTAMP

Bit 16 **TAMP1INSEL**: TAMPER1 mapping
0: RTC_AF1 used as TAMPER1
1: RTC_AF2 used as TAMPER1
Note: TAMP1E must be reset when TAMP1INSEL is changed to avoid unwanted setting of TAMP1F.

Bit 15 **TAMPPUDIS**: TAMPER pull-up disable
This bit determines if each of the tamper pins are pre-charged before each sample.
0: Precharge tamper pins before sampling (enable internal pull-up)
1: Disable precharge of tamper pins
Note:

Bits 14:13 **TAMPPRCH[1:0]**: Tamper precharge duration
These bit determine the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the tamper inputs.
0x0: 1 RTCCLK cycle
0x1: 2 RTCCLK cycles
0x2: 4 RTCCLK cycles
0x3: 8 RTCCLK cycles

Bits 12:11 **TAMPFLT[1:0]**: Tamper filter count

These bits determines the number of consecutive samples at the specified level (TAMP*TRG) necessary to activate a Tamper event. TAMPFLT is valid for each of the tamper inputs.

0x0: Tamper is activated on edge of tamper input transitions to the active level (no internal pull-up on tamper input).

0x1: Tamper is activated after 2 consecutive samples at the active level.

0x2: Tamper is activated after 4 consecutive samples at the active level.

0x3: Tamper is activated after 8 consecutive samples at the active level.

Bits 10:8 **TAMPFREQ[2:0]**: Tamper sampling frequency

Determines the frequency at which each of the tamper inputs are sampled.

0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)

0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)

0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)

0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)

0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)

0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)

0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)

0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

Bit 7 **TAMPTS**: Activate timestamp on tamper detection event

0: Tamper detection event does not cause a timestamp to be saved

1: Save timestamp on tamper detection event

TAMPTS is valid even if TSE=0 in the RTC_CR register.

Bits 6:5 Reserved. Always read as 0.

Bit 4 **TAMP2TRG**: Active level for tamper 2

if TAMPFLT != 00

0: TAMPER2 staying low triggers a tamper detection event.

1: TAMPER2 staying high triggers a tamper detection event.

if TAMPFLT = 00:

0: TAMPER2 rising edge triggers a tamper detection event.

1: TAMPER2 falling edge triggers a tamper detection event.

Bit 3 **TAMP2E**: Tamper 2 detection enable

0: Tamper 2 detection disabled

1: Tamper 2 detection enabled

Bit 2 **TAMPIE**: Tamper interrupt enable

0: Tamper interrupt disabled

1: Tamper interrupt enabled

Bit 1 **TAMP1TRG**: Active level for tamper 1

if TAMPFLT != 00

0: TAMPER1 staying low triggers a tamper detection event.

1: TAMPER1 staying high triggers a tamper detection event.

if TAMPFLT = 00:

0: TAMPER1 rising edge triggers a tamper detection event.

1: TAMPER1 falling edge triggers a tamper detection event.

Caution: When TAMPFLT = 0, TAMPxE must be reset when TAMPxTRG is changed to avoid spuriously setting TAMPxF.

Bit 0 **TAMP1E**: Tamper 1 detection enable

0: Tamper 1 detection disabled

1: Tamper 1 detection enabled

26.6.18 RTC alarm A sub second register (RTC_ALRMASSR)

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
Reserved				MASKSS[3:0]				Reserved							
r	r	r	r	rw	rw	rw	rw	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SS[14:0]														
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:28 Reserved

Bits 27:24 **MASKSS[3:0]**: Mask the most-significant bits starting at this bit

0: No comparison on sub seconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[14:1] are don't care in Alarm A comparison. Only SS[0] is compared.

2: SS[14:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.

3: SS[14:3] are don't care in Alarm A comparison. Only SS[2:0] are compared.

...

12: SS[14:12] are don't care in Alarm A comparison. SS[11:0] are compared.

13: SS[14:13] are don't care in Alarm A comparison. SS[12:0] are compared.

14: SS[14] is don't care in Alarm A comparison. SS[13:0] are compared.

15: All 15 SS bits are compared and must match to activate alarm.

The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved

Bits 14:0 **SS[14:0]**: Sub seconds value

This value is compared with the contents of the synchronous prescaler's counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

Note: This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in RTC register write protection on page 803

26.6.19 RTC alarm B sub second register (RTC_ALRMBSSR)

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
Reserved				MASKSS[3:0]				Reserved							
r	r	r	r	rw	rw	rw	rw	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SS[14:0]														
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:28 Reserved

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

Bits 14:0 **SS[14:0]**: Sub seconds value

This value is compared with the contents of the synchronous prescaler's counter to determine if Alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

Note: This register can be written only when ALRBIE 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

26.6.20 RTC backup registers (RTC_BKPxR)

Address offset: 0x50 to 0x9C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:0 BKP[31:0]

The application can write or read data to and from these registers.
They are powered-on by V_{BAT} when V_{DD} is switched off, so that they are not reset by System reset, and their contents remain valid when the device operates in low-power mode.
This register is reset on a tamper detection event, as long as TAMPxF=1.

26.6.21 RTC register map

Table 121. RTC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**RTC_TR**	Reserved									PM	HT [1:0]		HU[3:0]				Reserved	MNT[2:0]			MNU[3:0]				Reserved	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**	Reserved									YT[3:0]			YU[3:0]				WDU[2:0]			MT	MU[3:0]				Reserved	DT [1:0]		DU[3:0]			
	Reset value																	0	0	1	0	0	0	0	1		0	0	0	0	0	0	1
0x08	**RTC_CR**	Reserved									COE	OSEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	DCE	FMT	BYPSHAD	REFCKON	TSEDGE	WCKSEL [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	Reserved																	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRBWF	ALRAWF
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
0x10	**RTC_PRER**	Reserved									PREDIV_A[6:0]							Reserved	PREDIV_S[14:0]														
	Reset value										1	1	1	1	1	1	1		0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	**RTC_WUTR**	Reserved																WUT[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	**RTC_CALIBR**	Reserved																								DCS	Reserved	DC[4:0]					
	Reset value																									0		0	0	0	0	0	

Table 121. RTC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C	**RTC_ALRMAR**	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**RTC_ALRMBR**	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK2	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**RTC_WPR**	Reserved																								KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	**RTC_SSR**	Reserved																SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	RTC_SHIFTR	ADD1S	Reserved															SUBFS[14:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	**RTC_TSTR**	Reserved								PM	HT[1:0]			HU[3:0]				Reserved	MNT[2:0]			MNU[3:0]				Reserved	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
0x38	**RTC_TSSSR**	Reserved																SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**RTC_ CALR**	Reserved																CALP	CALW8	CALW16	Reserved					CALM[8:0]							
	Reset value																	0	0	0						0	0	0	0	0	0	0	0
0x40	RTC_TAFCR	Reserved													ALARMOUTTYPE	TSINSEL	TAMP1INSEL	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMPTS	Reserved	TAMP2TRG	TAMP2E	TAMPIE	TAMP1ETRG	TAMP1E	
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	
0x44	**RTC_ ALRMASSR**	Reserved				MASKSS[3:0]				Reserved									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**	Reserved				MASKSS[3:0]				Reserved									SS[14:0]														
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50 to 0x9C	**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
	to **RTC_BKP19R**	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory map* for the register boundary addresses.

Caution: In *Table 121*, the reset value is the value after a backup domain reset. The majority of the registers are not affected by a system reset. For more information, please refer to *Section 26.3.7: Resetting the RTC*.

27 Inter-integrated circuit (I2C) interface

This section applies to the whole STM32F4xx family, unless otherwise specified.

27.1 I²C introduction

I²C (inter-integrated circuit) bus Interface serves as an interface between the microcontroller and the serial I²C bus. It provides multimaster capability, and controls all I²C bus-specific sequencing, protocol, arbitration and timing. It supports the standard mode (Sm, up to 100 kHz) and Fm mode (Fm, up to 400 kHz).

It may be used for a variety of purposes, including CRC generation and verification, SMBus (system management bus) and PMBus (power management bus).

Depending on specific device implementation DMA capability can be available for reduced CPU overload.

27.2 I²C main features

- Parallel-bus/I²C protocol converter
- Multimaster capability: the same interface can act as Master or Slave
- I²C Master features:
  - Clock generation
  - Start and Stop generation
- I²C Slave features:
  - Programmable I²C Address detection
  - Dual Addressing Capability to acknowledge 2 slave addresses
  - Stop bit detection
- Generation and detection of 7-bit/10-bit addressing and General Call
- Supports different communication speeds:
  - Standard Speed (up to 100 kHz)
  - Fast Speed (up to 400 kHz)
- Analog noise filter
- Programmable digital noise filter for STM32F42xxx and STM32F43xxx
- Status flags:
  - Transmitter/Receiver mode flag
  - End-of-Byte transmission flag
  - I²C busy flag
- Error flags:
  - Arbitration lost condition for master mode
  - Acknowledgment failure after address/ data transmission
  - Detection of misplaced start or stop condition
  - Overrun/Underrun if clock stretching is disabled
- 2 Interrupt vectors:

  - 1 Interrupt for successful address/ data communication
  - 1 Interrupt for error condition
- Optional clock stretching
- 1-byte buffer with DMA capability
- Configurable PEC (packet error checking) generation or verification:
  - PEC value can be transmitted as last byte in Tx mode
  - PEC error checking for last received byte
- SMBus 2.0 Compatibility:
  - 25 ms clock low timeout delay
  - 10 ms master cumulative clock low extend time
  - 25 ms slave cumulative clock low extend time
  - Hardware PEC generation/verification with ACK control
  - Address Resolution Protocol (ARP) supported
- PMBus Compatibility

Note: Some of the above features may not be available in certain products. The user should refer to the product data sheet, to identify the specific features supported by the I²C interface implementation.

27.3 I²C functional description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I²C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz) or fast (up to 400 kHz) I²C bus.

27.3.1 Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master, after it generates a START condition and from master to slave, if an arbitration loss or a Stop generation occurs, allowing multimaster capability.

Communication flow

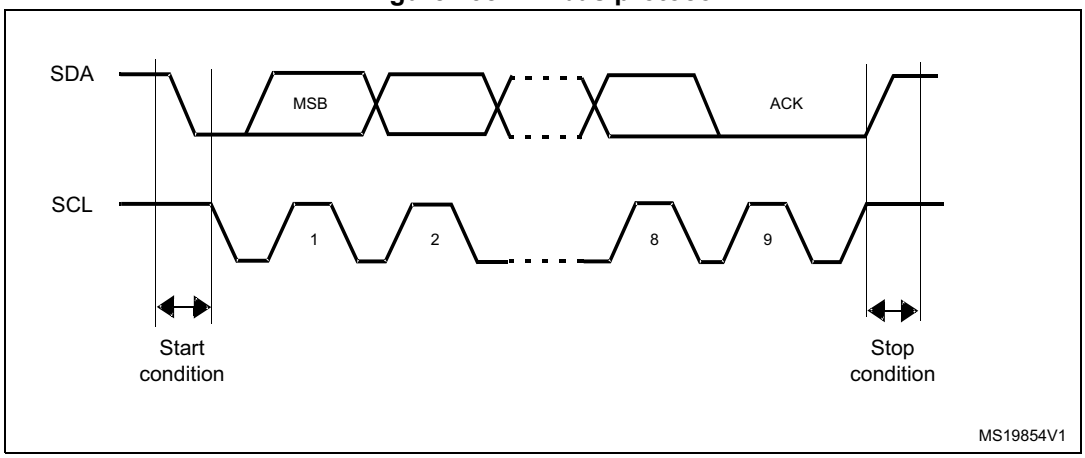
In Master mode, the I²C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a start condition and ends with a stop condition. Both start and stop conditions are generated in master mode by software.

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 may be enabled or disabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the start condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to *Figure 238*.

Figure 238. I²C bus protocol



Acknowledge may be enabled or disabled by software. The I²C interface addresses (dual addressing 7-bit/ 10-bit and/or general call address) can be selected by software.

The block diagram of the I²C interface is shown in *Figure 239*.

Figure 239. I²C block diagram for STM32F40x/41x

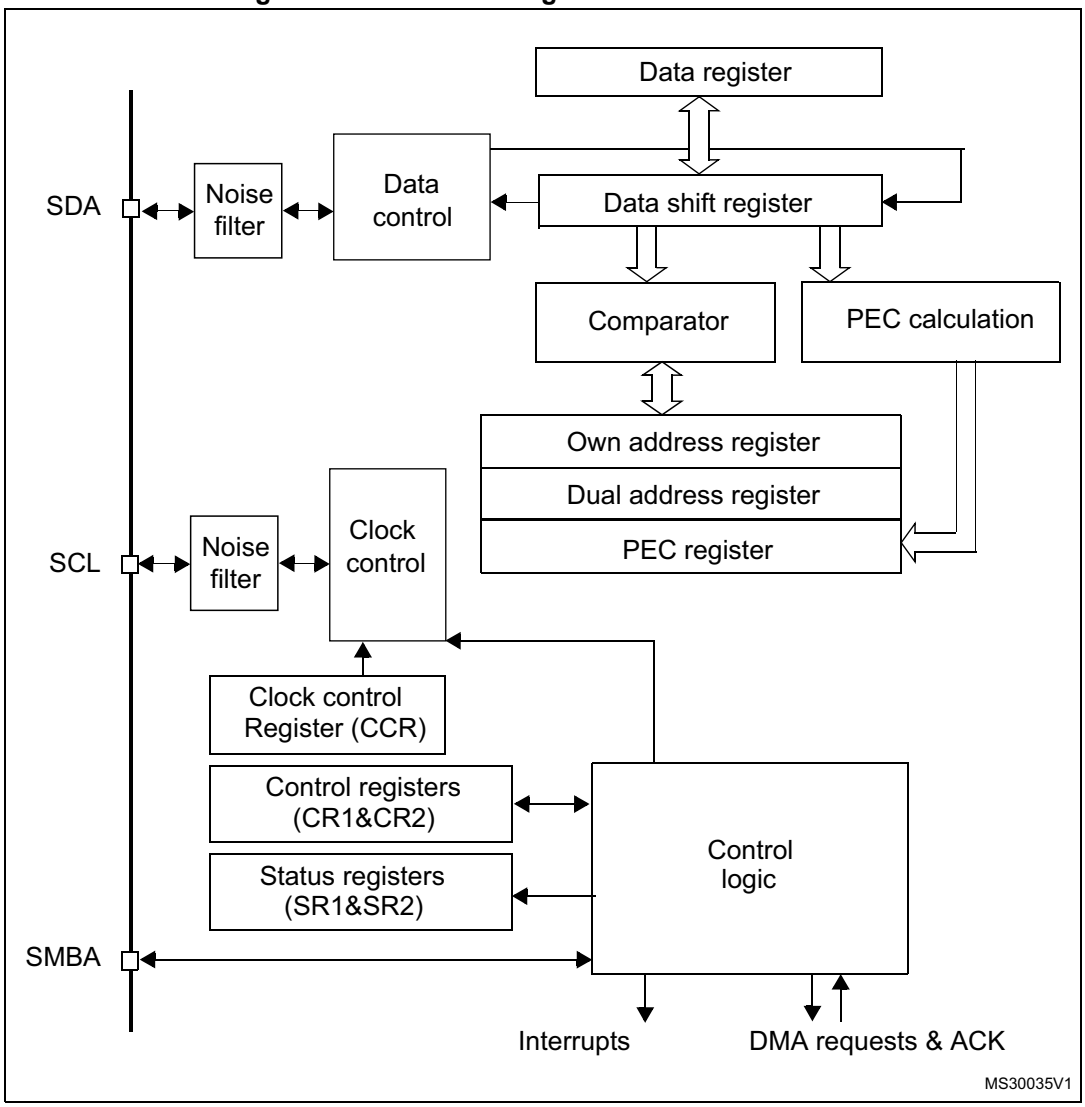
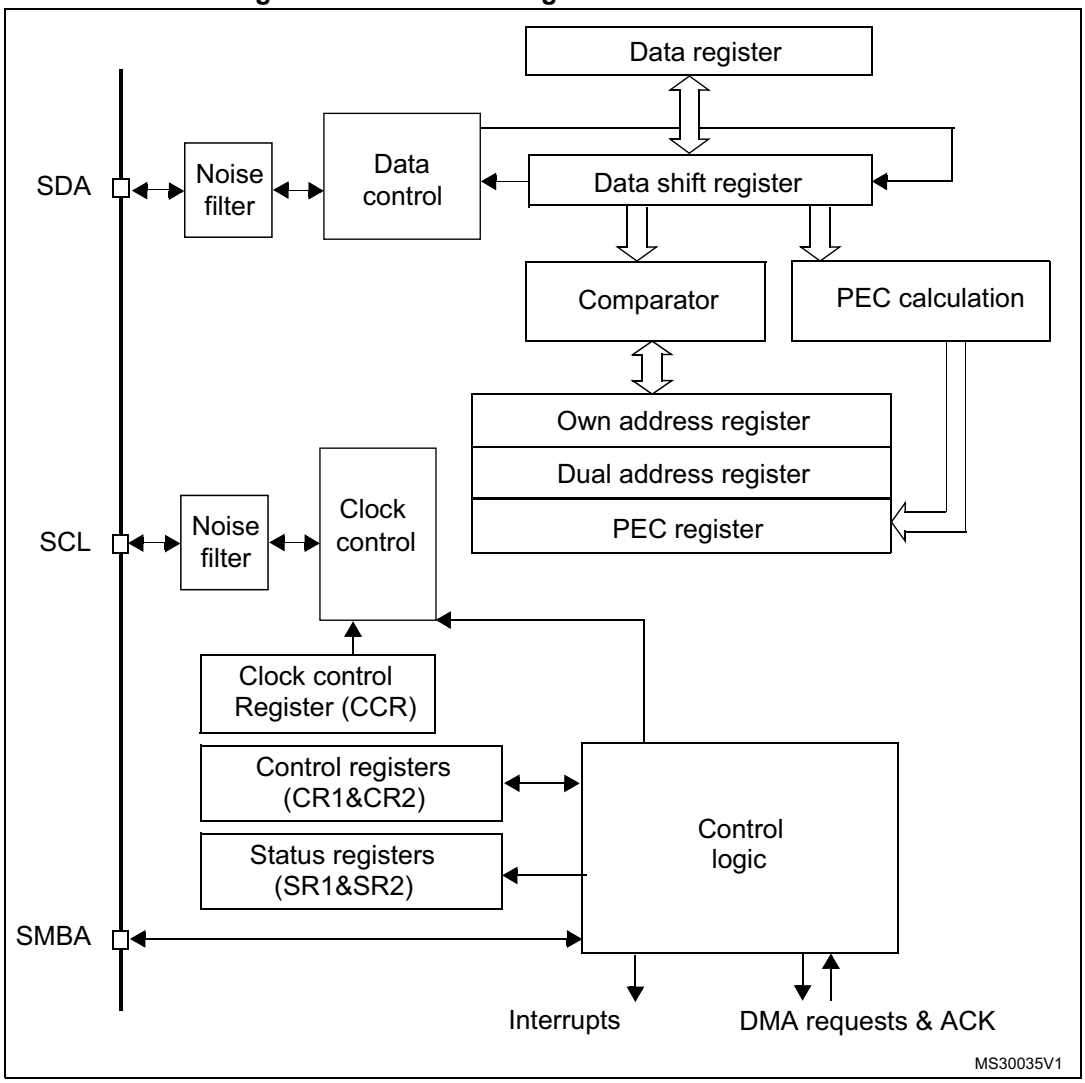


Figure 240. I²C block diagram for STM32F42x/43x



1. SMBA is an optional signal in SMBus mode. This signal is not applicable if SMBus is disabled.

27.3.2 I²C slave mode

By default the I²C interface operates in Slave mode. To switch from default Slave mode to Master mode a Start condition generation is needed.

The peripheral input clock must be programmed in the I2C_CR2 register in order to generate correct timings. The peripheral input clock frequency must be at least:

- 2 MHz in Sm mode
- 4 MHz in Fm mode

As soon as a start condition is detected, the address is received from the SDA line and sent to the shift register. Then it is compared with the address of the interface (OAR1) and with OAR2 (if ENDUAL=1) or the General Call address (if ENGC = 1).

Note: In 10-bit addressing mode, the comparison includes the header sequence (11110xx0), where xx denotes the two most significant bits of the address.

Header or address not matched: the interface ignores it and waits for another Start condition.

Header matched (10-bit mode only): the interface generates an acknowledge pulse if the ACK bit is set and waits for the 8-bit slave address.

Address matched: the interface generates in sequence:

- An acknowledge pulse if the ACK bit is set
- The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.
- If ENDUAL=1, the software has to read the DUALF bit to check which slave address has been acknowledged.

In 10-bit mode, after receiving the address sequence the slave is always in Receiver mode. It will enter Transmitter mode on receiving a repeated Start condition followed by the header sequence with matching address bits and the least significant bit set (11110xx1).

The TRA bit indicates whether the slave is in Receiver or Transmitter mode.

Slave transmitter

Following the address reception and after clearing ADDR, the slave sends bytes from the DR register to the SDA line via the internal shift register.

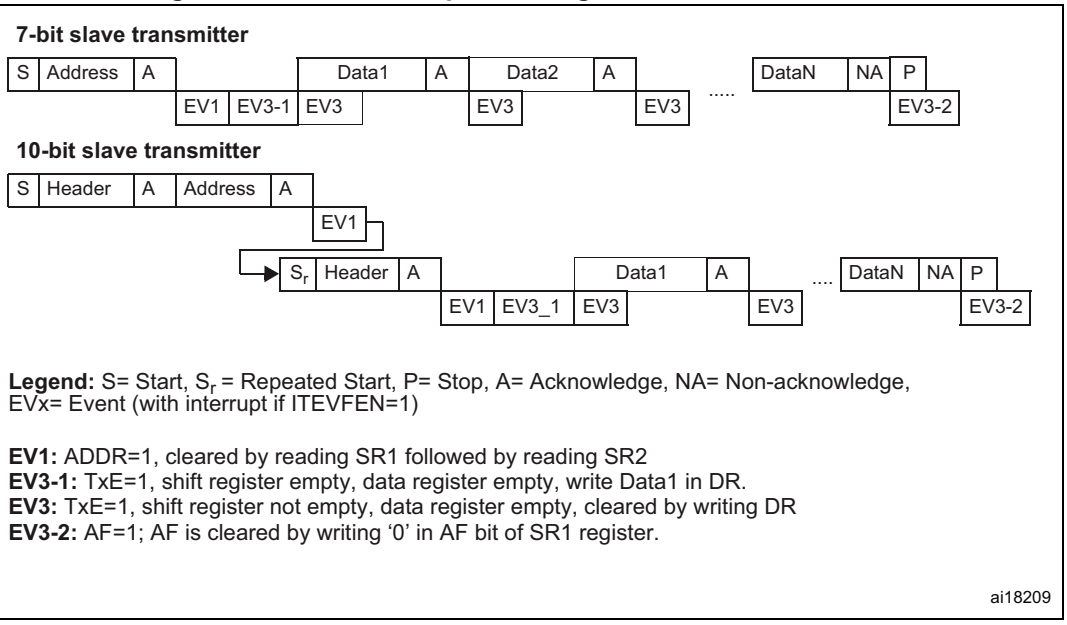
The slave stretches SCL low until ADDR is cleared and DR filled with the data to be sent (see *Figure 241* Transfer sequencing EV1 EV3).

When the acknowledge pulse is received:

- The TxE bit is set by hardware with an interrupt if the ITEVFEN and the ITBUFEN bits are set.

If TxE is set and some data were not written in the I2C_DR register before the end of the next data transmission, the BTF bit is set and the interface waits until BTF is cleared by a read to I2C_SR1 followed by a write to the I2C_DR register, stretching SCL low.

Figure 241. Transfer sequence diagram for slave transmitter



Legend: S= Start, S_r = Repeated Start, P= Stop, A= Acknowledge, NA= Non-acknowledge, EVx= Event (with interrupt if ITEVFEN=1)

EV1: ADDR=1, cleared by reading SR1 followed by reading SR2
EV3-1: TxE=1, shift register empty, data register empty, write Data1 in DR.
EV3: TxE=1, shift register not empty, data register empty, cleared by writing DR
EV3-2: AF=1; AF is cleared by writing '0' in AF bit of SR1 register.

ai18209

1. The EV1 and EV3_1 events stretch SCL low until the end of the corresponding software sequence.
2. The EV3 event stretches SCL low if the software sequence is not completed before the end of the next byte transmission.

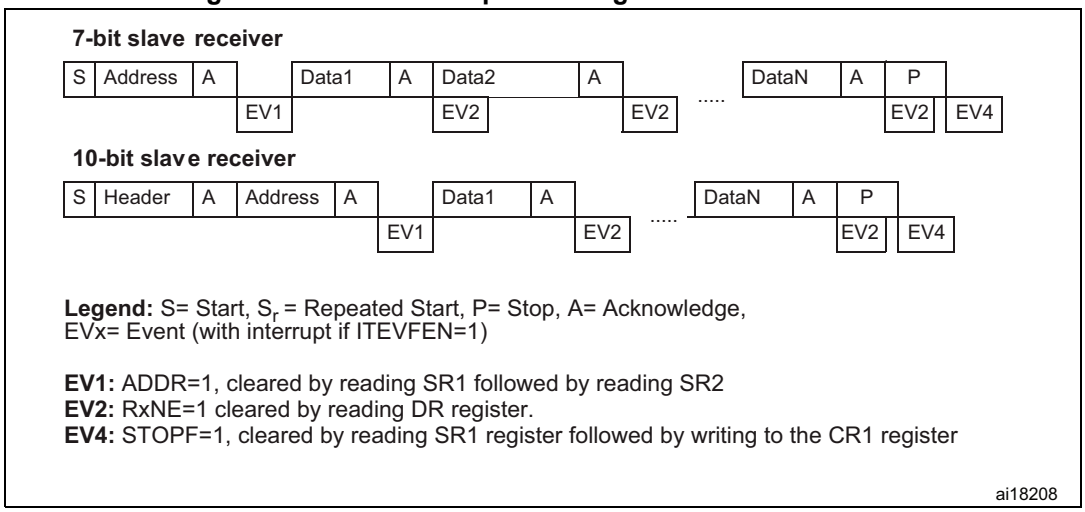
Slave receiver

Following the address reception and after clearing ADDR, the slave receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

- An acknowledge pulse if the ACK bit is set
- The RxNE bit is set by hardware and an interrupt is generated if the ITEVFEN and ITBUFEN bit is set.

If RxNE is set and the data in the DR register is not read before the end of the next data reception, the BTF bit is set and the interface waits until BTF is cleared by a read from the I2C_DR register, stretching SCL low (see *Figure 242* Transfer sequencing).

Figure 242. Transfer sequence diagram for slave receiver



1. The EV1 event stretches SCL low until the end of the corresponding software sequence.
2. The EV2 event stretches SCL low if the software sequence is not completed before the end of the next byte reception.
3. After checking the SR1 register content, the user should perform the complete clearing sequence for each flag found set.
   Thus, for ADDR and STOPF flags, the following sequence is required inside the I2C interrupt routine:
   READ SR1
   if (ADDR == 1) {READ SR1; READ SR2}
   if (STOPF == 1) {READ SR1; WRITE CR1}
   The purpose is to make sure that both ADDR and STOPF flags are cleared if both are found set.

Closing slave communication

After the last data byte is transferred a Stop Condition is generated by the master. The interface detects this condition and sets:

- The STOPF bit and generates an interrupt if the ITEVFEN bit is set.

The STOPF bit is cleared by a read of the SR1 register followed by a write to the CR1 register (see EV4 in *Figure 242*).

27.3.3 I²C master mode

In Master mode, the I²C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a Start condition and ends with a Stop condition. Master mode is selected as soon as the Start condition is generated on the bus with a START bit.

The following is the required sequence in master mode.

- Program the peripheral input clock in I2C_CR2 Register in order to generate correct timings
- Configure the clock control registers
- Configure the rise time register
- Program the I2C_CR1 register to enable the peripheral
- Set the START bit in the I2C_CR1 register to generate a Start condition

The peripheral input clock frequency must be at least:

- 2 MHz in Sm mode
- 4 MHz in Fm mode

SCL master clock generation

The CCR bits are used to generate the high and low level of the SCL clock, starting from the generation of the rising and falling edge (respectively). As a slave may stretch the SCL line, the peripheral checks the SCL input from the bus at the end of the time programmed in TRISE bits after rising edge generation.

- If the SCL line is low, it means that a slave is stretching the bus, and the high level counter stops until the SCL line is detected high. This allows to guarantee the minimum HIGH period of the SCL clock parameter.
- If the SCL line is high, the high level counter keeps on counting.

Indeed, the feedback loop from the SCL rising edge generation by the peripheral to the SCL rising edge detection by the peripheral takes time even if no slave stretches the clock. This loopback duration is linked to the SCL rising time (impacting SCL VIH input detection), plus delay due to the noise filter present on the SCL input path, plus delay due to internal SCL input synchronization with APB clock. The maximum time used by the feedback loop is programmed in the TRISE bits, so that the SCL frequency remains stable whatever the SCL rising time.

Start condition

Setting the START bit causes the interface to generate a Start condition and to switch to Master mode (MSL bit set) when the BUSY bit is cleared.

Note: In master mode, setting the START bit causes the interface to generate a ReStart condition at the end of the current byte transfer.

Once the Start condition is sent:

- The SB bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

Then the master waits for a read of the SR1 register followed by a write in the DR register with the Slave address (see *Figure 243* and *Figure 244* Transfer sequencing EV5).

Slave address transmission

Then the slave address is sent to the SDA line via the internal shift register.

- In 10-bit addressing mode, sending the header sequence causes the following event:
  - The ADD10 bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

  Then the master waits for a read of the SR1 register followed by a write in the DR register with the second address byte (see *Figure 243* and *Figure 244* Transfer sequencing).
  - The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

  Then the master waits for a read of the SR1 register followed by a read of the SR2 register (see *Figure 243* and *Figure 244* Transfer sequencing).
- In 7-bit addressing mode, one address byte is sent.

  As soon as the address byte is sent,
  - The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

  Then the master waits for a read of the SR1 register followed by a read of the SR2 register (see *Figure 243* and *Figure 244* Transfer sequencing).

The master can decide to enter Transmitter or Receiver mode depending on the LSB of the slave address sent.

- In 7-bit addressing mode,
  - To enter Transmitter mode, a master sends the slave address with LSB reset.
  - To enter Receiver mode, a master sends the slave address with LSB set.
- In 10-bit addressing mode,
  - To enter Transmitter mode, a master sends the header (11110xx0) and then the slave address, (where xx denotes the two most significant bits of the address).
  - To enter Receiver mode, a master sends the header (11110xx0) and then the slave address. Then it should send a repeated Start condition followed by the header (11110xx1), (where xx denotes the two most significant bits of the address).

  The TRA bit indicates whether the master is in Receiver or Transmitter mode.

Master transmitter

Following the address transmission and after clearing ADDR, the master sends bytes from the DR register to the SDA line via the internal shift register.

The master waits until the first data byte is written into I2C_DR (see *Figure 243* Transfer sequencing EV8_1).

When the acknowledge pulse is received, the TxE bit is set by hardware and an interrupt is generated if the ITEVFEN and ITBUFEN bits are set.

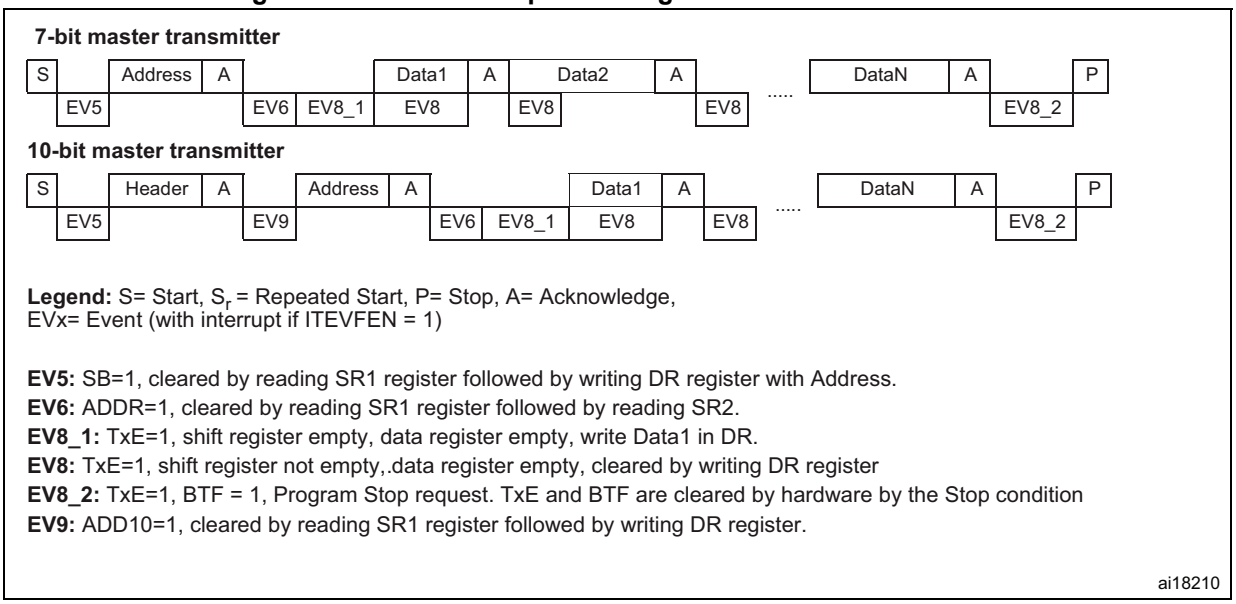
If TxE is set and a data byte was not written in the DR register before the end of the last data transmission, BTF is set and the interface waits until BTF is cleared by a write to I2C_DR, stretching SCL low.

Closing the communication

After the last byte is written to the DR register, the STOP bit is set by software to generate a Stop condition (see *Figure 243* Transfer sequencing EV8_2). The interface automatically goes back to slave mode (MSL bit cleared).

Note: Stop condition should be programmed during EV8_2 event, when either TxE or BTF is set.

Figure 243. Transfer sequence diagram for master transmitter



1. The EV5, EV6, EV9, EV8_1 and EV8_2 events stretch SCL low until the end of the corresponding software sequence.
2. The EV8 event stretches SCL low if the software sequence is not complete before the end of the next byte transmission.

Master receiver

Following the address transmission and after clearing ADDR, the I^2C interface enters Master Receiver mode. In this mode the interface receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

1. An acknowledge pulse if the ACK bit is set
2. The RxNE bit is set and an interrupt is generated if the ITEVFEN and ITBUFEN bits are set (see *Figure 244* Transfer sequencing EV7).

If the RxNE bit is set and the data in the DR register is not read before the end of the last data reception, the BTF bit is set by hardware and the interface waits until BTF is cleared by a read in the DR register, stretching SCL low.

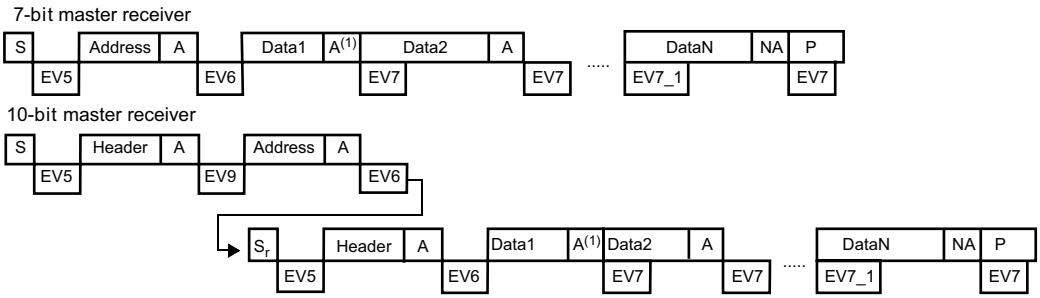
Closing the communication

The master sends a NACK for the last byte received from the slave. After receiving this NACK, the slave releases the control of the SCL and SDA lines. Then the master can send a Stop/Restart condition.

1. To generate the nonacknowledge pulse after the last received data byte, the ACK bit must be cleared just after reading the second last data byte (after second last RxNE event).
2. In order to generate the Stop/Restart condition, software must set the STOP/START bit after reading the second last data byte (after the second last RxNE event).
3. In case a single byte has to be received, the Acknowledge disable is made during EV6 (before ADDR flag is cleared) and the STOP condition generation is made after EV6.

After the Stop condition generation, the interface goes automatically back to slave mode (MSL bit cleared).

Figure 244. Transfer sequence diagram for master receiver



Legend: S= Start, S_r = Repeated Start, P= Stop, A= Acknowledge, NA= Non-acknowledge,
EVx= Event (with interrupt if ITEVFEN=1)

EV5: SB=1, cleared by reading SR1 register followed by writing DR register.

EV6: ADDR=1, cleared by reading SR1 register followed by reading SR2. In 10-bit master receiver mode, this sequence should be followed by writing CR2 with START = 1.
In case of the reception of 1 byte, the Acknowledge disable must be performed during EV6 event, i.e. before clearing ADDR flag.

EV7: RxNE=1 cleared by reading DR register.

EV7_1: RxNE=1 cleared by reading DR register, program ACK=0 and STOP request

EV9: ADD10=1, cleared by reading SR1 register followed by writing DR register.

ai17540d

1. If a single byte is received, it is NA.
2. The EV5, EV6 and EV9 events stretch SCL low until the end of the corresponding software sequence.
3. The EV7 event stretches SCL low if the software sequence is not completed before the end of the next byte reception.
4. The EV7_1 software sequence must be completed before the ACK pulse of the current byte transfer.

The procedures described below are recommended if the EV7-1 software sequence is not completed before the ACK pulse of the current byte transfer.

These procedures must be followed to make sure:

- The ACK bit is set low on time before the end of the last data reception
- The STOP bit is set high after the last data reception without reception of supplementary data.

For 2-byte reception:

- Wait until ADDR = 1 (SCL stretched low until the ADDR flag is cleared)
- Set ACK low, set POS high
- Clear ADDR flag
- Wait until BTF = 1 (Data 1 in DR, Data2 in shift register, SCL stretched low until a data 1 is read)
- Set STOP high
- Read data 1 and 2

For N >2 -byte reception, from N-2 data reception

- Wait until BTF = 1 (data N-2 in DR, data N-1 in shift register, SCL stretched low until data N-2 is read)
- Set ACK low
- Read data N-2
- Wait until BTF = 1 (data N-1 in DR, data N in shift register, SCL stretched low until a data N-1 is read)
- Set STOP high
- Read data N-1 and N

27.3.4 Error conditions

The following are the error conditions which may cause communication to fail.

Bus error (BERR)

This error occurs when the I^2C interface detects an external Stop or Start condition during an address or a data transfer. In this case:

- the BERR bit is set and an interrupt is generated if the ITERREN bit is set
- in Slave mode: data are discarded and the lines are released by hardware:
  - in case of a misplaced Start, the slave considers it is a restart and waits for an address, or a Stop condition
  - in case of a misplaced Stop, the slave behaves like for a Stop condition and the lines are released by hardware
- In Master mode: the lines are not released and the state of the current transmission is not affected. It is up to the software to abort or not the current transmission

Acknowledge failure (AF)

This error occurs when the interface detects a nonacknowledge bit. In this case:

- the AF bit is set and an interrupt is generated if the ITERREN bit is set
- a transmitter which receives a NACK must reset the communication:
  - If Slave: lines are released by hardware
  - If Master: a Stop or repeated Start condition must be generated by software

Arbitration lost (ARLO)

This error occurs when the I^2C interface detects an arbitration lost condition. In this case

- the ARLO bit is set by hardware (and an interrupt is generated if the ITERREN bit is set)
- the I^2C Interface goes automatically back to slave mode (the MSL bit is cleared). When the I^2C loses the arbitration, it is not able to acknowledge its slave address in the same transfer, but it can acknowledge it after a repeated Start from the winning master.
- lines are released by hardware

Overrun/underrun error (OVR)

An overrun error can occur in slave mode when clock stretching is disabled and the I^2C interface is receiving data. The interface has received a byte (RxNE=1) and the data in DR has not been read, before the next byte is received by the interface. In this case,

- The last received byte is lost.
- In case of Overrun error, software should clear the RxNE bit and the transmitter should re-transmit the last received byte.

Underrun error can occur in slave mode when clock stretching is disabled and the I^2C interface is transmitting data. The interface has not updated the DR with the next byte (TxE=1), before the clock comes for the next byte. In this case,

- The same byte in the DR register will be sent again
- The user should make sure that data received on the receiver side during an underrun error are discarded and that the next bytes are written within the clock low time specified in the I^2C bus standard.

For the first byte to be transmitted, the DR must be written after ADDR is cleared and before the first SCL rising edge. If not possible, the receiver must discard the first data.

27.3.5 Programmable noise filter

The programmable noise filter is available on STM32F42xxx and STM32F43xxx devices only.

In Fm mode, the I^2C standard requires that spikes are suppressed to a length of 50 ns on SDA and SCL lines.

An analog noise filter is implemented in the SDA and SCL I/Os. This filter is enabled by default and can be disabled by setting the ANOFF bit in the I2C_FLTR register.

A digital noise filter can be enabled by configuring the DNF[3:0] bits to a non-zero value. This suppresses the spikes on SDA and SCL inputs with a length of up to DNF[3:0] * T_{PCLK1}.

Enabling the digital noise filter increases the SDA hold time by (DNF[3:0] +1)* T_{PCLK}.

To be compliant with the maximum hold time of the I^2C-bus specification version 2.1 (Thd:dat), the DNF bits must be programmed using the constraints shown in *Table 122*, and assuming that the analog filter is disabled.

Note: DNF[3:0] must only be configured when the I^2C is disabled (PE = 0). If the analog filter is also enabled, the digital filter is added to the analog filter.

Table 122. Maximum DNF[3:0] value to be compliant with Thd:dat(max)

PCLK1 frequency	Maximum DNF value	
	Sm mode	Fm mode
2 <= F_{PCLK1} <= 5	2	0
5 < F_{PCLK1} <= 10	12	0
10 < F_{PCLK1} <= 20	15	1
20 < F_{PCLK1} <= 30	15	7

Table 122. Maximum DNF[3:0] value to be compliant with Thd:dat(max) (continued)

PCLK1 frequency	Maximum DNF value	
	Sm mode	Fm mode
30 < F_{PCLK1} <= 40	15	13
40 < F_{PCLK1} <= 50	15	15

Note: For each frequency range, the constraint is given based on the worst case which is the minimum frequency of the range. Greater DNF values can be used if the system can support maximum hold time violation.

27.3.6 SDA/SCL line control

- If clock stretching is enabled:
  - Transmitter mode: If TxE=1 and BTF=1: the interface holds the clock line low before transmission to wait for the microcontroller to write the byte in the Data Register (both buffer and shift register are empty).
  - Receiver mode: If RxNE=1 and BTF=1: the interface holds the clock line low after reception to wait for the microcontroller to read the byte in the Data Register (both buffer and shift register are full).
- If clock stretching is disabled in Slave mode:
  - Overrun Error in case of RxNE=1 and no read of DR has been done before the next byte is received. The last received byte is lost.
  - Underrun Error in case TxE=1 and no write into DR has been done before the next byte must be transmitted. The same byte will be sent again.
  - Write Collision not managed.

27.3.7 SMBus

Introduction

The System Management Bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I^2C principles of operation. SMBus provides a control bus for system and power management related tasks. A system may use SMBus to pass messages to and from devices instead of toggling individual control lines.

The System Management Bus Specification refers to three types of devices. A *slave* is a device that is receiving or responding to a command. A *master* is a device that issues commands, generates the clocks, and terminates the transfer. A *host* is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

Similarities between SMBus and I^2C

- 2-wire bus protocol (1 Clk, 1 Data) + SMBus Alert line optional
- Master-slave communication, Master provides clock
- Multi master capability
- SMBus data format similar to I^2C 7-bit addressing format (*Figure 238*).

Differences between SMBus and I^2C

The following table describes the differences between SMBus and I^2C.

Table 123. SMBus vs. I^2C

SMBus	I^2C
Max. speed 100 kHz	Max. speed 400 kHz
Min. clock speed 10 kHz	No minimum clock speed
35 ms clock low timeout	No timeout
Logic levels are fixed	Logic levels are V_{DD} dependent
Different address types (reserved, dynamic etc.)	7-bit, 10-bit and general call slave address types
Different bus protocols (quick command, process call etc.)	No bus protocols

SMBus application usage

With System Management Bus, a device can provide manufacturer information, tell the system what its model/part number is, save its state for a suspend event, report different types of errors, accept control parameters, and return its status. SMBus provides a control bus for system and power management related tasks.

Device identification

Any device that exists on the System Management Bus as a slave has a unique address called the Slave Address. For the list of reserved slave addresses, refer to the SMBus specification version. 2.0 (*http://smbus.org/*).

Bus protocols

The SMBus specification supports up to nine bus protocols. For more details of these protocols and SMBus address types, refer to SMBus specification version. 2.0. These protocols should be implemented by the user software.

Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. The Address Resolution Protocol (ARP) has the following attributes:

- Address assignment uses the standard SMBus physical layer arbitration mechanism
- Assigned addresses remain constant while device power is applied; address retention through device power loss is also allowed
- No additional SMBus packet overhead is incurred after address assignment. (i.e. subsequent accesses to assigned slave addresses have the same overhead as accesses to fixed address devices.)
- Any SMBus master can enumerate the bus

Unique device identifier (UDID)

In order to provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique device identifier (UDID).

For the details on 128-bit UDID and more information on ARP, refer to SMBus specification version 2.0.

SMBus alert mode

SMBus Alert is an optional signal with an interrupt line for devices that want to trade their ability to master for a pin. SMBA is a wired-AND signal just as the SCL and SDA signals are. SMBA is used in conjunction with the SMBus General Call Address. Messages invoked with the SMBus are two bytes long.

A slave-only device can signal the host through SMBA that it wants to talk by setting ALERT bit in I2C_CR1 register. The host processes the interrupt and simultaneously accesses all SMBA devices through the *Alert Response Address* (known as ARA having a value 0001 100X). Only the device(s) which pulled SMBA low will acknowledge the Alert Response Address. This status is identified using SMBALERT Status flag in I2C_SR1 register. The host performs a modified Receive Byte operation. The 7 bit device address provided by the slave transmit device is placed in the 7 most significant bits of the byte. The eighth bit can be a zero or one.

If more than one device pulls SMBA low, the highest priority (lowest address) device will win communication rights via standard arbitration during the slave address transfer. After acknowledging the slave address the device must disengage its SMBA pull-down. If the host still sees SMBA low when the message transfer is complete, it knows to read the ARA again.
A host which does not implement the SMBA signal may periodically access the ARA.

For more details on SMBus Alert mode, refer to SMBus specification version 2.0.

Timeout error

There are differences in the timing specifications between I^2C and SMBus.
SMBus defines a clock low timeout, TIMEOUT of 35 ms. Also SMBus specifies TLOW:SEXT as the cumulative clock low extend time for a slave device. SMBus specifies TLOW:MEXT as the cumulative clock low extend time for a master device. For more details on these timeouts, refer to SMBus specification version 2.0.

The status flag Timeout or Tlow Error in I2C_SR1 shows the status of this feature.

How to use the interface in SMBus mode

To switch from I^2C mode to SMBus mode, the following sequence should be performed.

- Set the SMBus bit in the I2C_CR1 register
- Configure the SMBTYPE and ENARP bits in the I2C_CR1 register as required for the application

If you want to configure the device as a master, follow the Start condition generation procedure in *Section 27.3.3*. Otherwise, follow the sequence in *Section 27.3.2*.

The application has to control the various SMBus protocols by software.

- SMB Device Default Address acknowledged if ENARP=1 and SMBTYPE=0
- SMB Host Header acknowledged if ENARP=1 and SMBTYPE=1
- SMB Alert Response Address acknowledged if SMBALERT=1

27.3.8 DMA requests

DMA requests (when enabled) are generated only for data transfer. DMA requests are generated by Data Register becoming empty in transmission and Data Register becoming full in reception. The DMA must be initialized and enabled before the I2C data transfer. The DMAEN bit must be set in the I2C_CR2 register before the ADDR event. In master mode or in slave mode when clock stretching is enabled, the DMAEN bit can also be set during the ADDR event, before clearing the ADDR flag. The DMA request must be served before the end of the current byte transfer. When the number of data transfers which has been programmed for the corresponding DMA stream is reached, the DMA controller sends an End of Transfer EOT signal to the I^2C interface and generates a Transfer Complete interrupt if enabled:

- Master transmitter: In the interrupt routine after the EOT interrupt, disable DMA requests then wait for a BTF event before programming the Stop condition.
- Master receiver
  - When the number of bytes to be received is equal to or greater than two, the DMA controller sends a hardware signal, EOT_1, corresponding to the last but one data byte (number_of_bytes – 1). If, in the I2C_CR2 register, the LAST bit is set, I^2C automatically sends a NACK after the next byte following EOT_1. The user can generate a Stop condition in the DMA Transfer Complete interrupt routine if enabled.
  - When a single byte must be received: the NACK must be programmed during EV6 event, i.e. program ACK=0 when ADDR=1, before clearing ADDR flag. Then the user can program the STOP condition either after clearing ADDR flag, or in the DMA Transfer Complete interrupt routine.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAEN bit in the I2C_CR2 register. Data will be loaded from a Memory area configured using the DMA peripheral (refer to the DMA specification) to the I2C_DR register whenever the TxE bit is set. To map a DMA stream x for I^2C transmission (where x is the stream number), perform the following sequence:

1. Set the I2C_DR register address in the DMA_SxPAR register. The data will be moved to this address from the memory after each TxE event.
2. Set the memory address in the DMA_SxMA0R register (and in DMA_SxMA1R register in the case of a bouble buffer mode). The data will be loaded into I2C_DR from this memory after each TxE event.
3. Configure the total number of bytes to be transferred in the DMA_SxNDTR register. After each TxE event, this value will be decremented.
4. Configure the DMA stream priority using the PL[0:1] bits in the DMA_SxCR register
5. Set the DIR bit in the DMA_SxCR register and configure interrupts after half transfer or full transfer depending on application requirements.
6. Activate the stream by setting the EN bit in the DMA_SxCR register.

When the number of data transfers which has been programmed in the DMA Controller registers is reached, the DMA controller sends an End of Transfer EOT/ EOT_1 signal to the I^2C interface and the DMA generates an interrupt, if enabled, on the DMA stream interrupt vector.

Note: Do not enable the ITBUFEN bit in the I2C_CR2 register if DMA is used for transmission.

Reception using DMA

DMA mode can be enabled for reception by setting the DMAEN bit in the I2C_CR2 register. Data will be loaded from the I2C_DR register to a Memory area configured using the DMA peripheral (refer to the DMA specification) whenever a data byte is received. To map a DMA stream x for I^2C reception (where x is the stream number), perform the following sequence:

1. Set the I2C_DR register address in DMA_SxPAR register. The data will be moved from this address to the memory after each RxNE event.
2. Set the memory address in the DMA_SxMA0R register (and in DMA_SxMA1R register in the case of a bouble buffer mode). The data will be loaded from the I2C_DR register to this memory area after each RxNE event.
3. Configure the total number of bytes to be transferred in the DMA_SxNDTR register. After each RxNE event, this value will be decremented.
4. Configure the stream priority using the PL[0:1] bits in the DMA_SxCR register
5. Reset the DIR bit and configure interrupts in the DMA_SxCR register after half transfer or full transfer depending on application requirements.
6. Activate the stream by setting the EN bit in the DMA_SxCR register.

When the number of data transfers which has been programmed in the DMA Controller registers is reached, the DMA controller sends an End of Transfer EOT/ EOT_1 signal to the I^2C interface and DMA generates an interrupt, if enabled, on the DMA stream interrupt vector.

Note: Do not enable the ITBUFEN bit in the I2C_CR2 register if DMA is used for reception.

27.3.9 Packet error checking

A PEC calculator has been implemented to improve the reliability of communication. The PEC is calculated by using the $C(x) = x^8 + x^2 + x + 1$ CRC-8 polynomial serially on each bit.

- PEC calculation is enabled by setting the ENPEC bit in the I2C_CR1 register. PEC is a CRC-8 calculated on all message bytes including addresses and R/W bits.
  - In transmission: set the PEC transfer bit in the I2C_CR1 register after the TxE event corresponding to the last byte. The PEC will be transferred after the last transmitted byte.
  - In reception: set the PEC bit in the I2C_CR1 register after the RxNE event corresponding to the last byte so that the receiver sends a NACK if the next received byte is not equal to the internally calculated PEC. In case of Master-Receiver, a NACK must follow the PEC whatever the check result. The PEC must

be set before the ACK of the CRC reception in slave mode. It must be set when the ACK is set low in master mode.

- A PECERR error flag/interrupt is also available in the I2C_SR1 register.
- If DMA and PEC calculation are both enabled:-
  - In transmission: when the I^2C interface receives an EOT signal from the DMA controller, it automatically sends a PEC after the last byte.
  - In reception: when the I^2C interface receives an EOT_1 signal from the DMA controller, it will automatically consider the next byte as a PEC and will check it. A DMA request is generated after PEC reception.
- To allow intermediate PEC transfers, a control bit is available in the I2C_CR2 register (LAST bit) to determine if it is really the last DMA transfer or not. If it is the last DMA request for a master receiver, a NACK is automatically sent after the last received byte.
- PEC calculation is corrupted by an arbitration loss.

27.4 I^2C interrupts

The table below gives the list of I^2C interrupt requests.

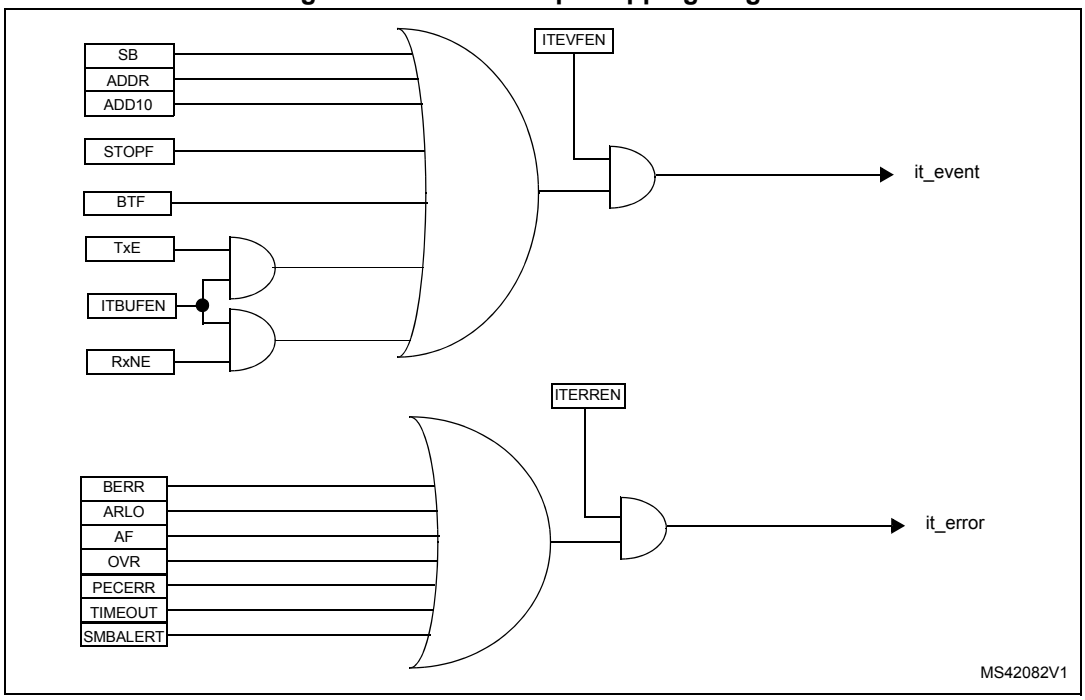
Table 124. I^2C Interrupt requests

Interrupt event	Event flag	Enable control bit
Start bit sent (Master)	SB	ITEVFEN
Address sent (Master) or Address matched (Slave)	ADDR	
10-bit header sent (Master)	ADD10	
Stop received (Slave)	STOPF	
Data byte transfer finished	BTF	
Receive buffer not empty	RxNE	ITEVFEN and ITBUFEN
Transmit buffer empty	TxE	
Bus error	BERR	ITERREN
Arbitration loss (Master)	ARLO	
Acknowledge failure	AF	
Overrun/Underrun	OVR	
PEC error	PECERR	
Timeout/Tlow error	TIMEOUT	
SMBus Alert	SMBALERT	

Note: SB, ADDR, ADD10, STOPF, BTF, RxNE and TxE are logically OR-ed on the same interrupt channel.

BERR, ARLO, AF, OVR, PECERR, TIMEOUT and SMBALERT are logically OR-ed on the same interrupt channel.

Figure 245. I²C interrupt mapping diagram



27.5 I²C debug mode

When the microcontroller enters the debug mode (Cortex®-M4 with FPU core halted), the SMBUS timeout either continues to work normally or stops, depending on the DBG_I2Cx_SMBUS_TIMEOUT configuration bits in the DBG module. For more details, refer to *Section 38.16.2: Debug support for timers, watchdog, bxCAN and I²C*.

27.6 I²C registers

Refer to *Section 1.1* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by half-words (16 bits) or words (32 bits).

27.6.1 I²C Control register 1 (I2C_CR1)

Address offset: 0x00
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	Res.	ALERT	PEC	POS	ACK	STOP	START	NO STRETCH	ENGC	ENPEC	ENARP	SMB TYPE	Res.	SMBUS	PE
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 15 **SWRST**: Software reset

When set, the I2C is under reset state. Before resetting this bit, make sure the I2C lines are released and the bus is free.

0: I²C Peripheral not under reset

1: I²C Peripheral under reset state

Note: This bit can be used to reinitialize the peripheral after an error or a locked state. As an example, if the BUSY bit is set and remains locked due to a glitch on the bus, the SWRST bit can be used to exit from this state.

Bit 14 Reserved, must be kept at reset value

Bit 13 **ALERT**: SMBus alert

This bit is set and cleared by software, and cleared by hardware when PE=0.

0: Releases SMBA pin high. Alert Response Address Header followed by NACK.

1: Drives SMBA pin low. Alert Response Address Header followed by ACK.

Bit 12 **PEC**: Packet error checking

This bit is set and cleared by software, and cleared by hardware when PEC is transferred or by a START or Stop condition or when PE=0.

0: No PEC transfer

1: PEC transfer (in Tx or Rx mode)

Note: PEC calculation is corrupted by an arbitration loss.

Bit 11 **POS**: Acknowledge/PEC Position (for data reception)

This bit is set and cleared by software and cleared by hardware when PE=0.

0: ACK bit controls the (N)ACK of the current byte being received in the shift register. The PEC bit indicates that current byte in shift register is a PEC.

1: ACK bit controls the (N)ACK of the next byte which will be received in the shift register. The PEC bit indicates that the next byte in the shift register is a PEC

Note: The POS bit must be used only in 2-byte reception configuration in master mode. It must be configured before data reception starts, as described in the 2-byte reception procedure recommended in Section : Master receiver on page 849.

Bit 10 **ACK**: Acknowledge enable
This bit is set and cleared by software and cleared by hardware when PE=0.
0: No acknowledge returned
1: Acknowledge returned after a byte is received (matched address or data)

Bit 9 **STOP**: Stop generation
The bit is set and cleared by software, cleared by hardware when a Stop condition is detected, set by hardware when a timeout error is detected.
In Master Mode:
0: No Stop generation.
1: Stop generation after the current byte transfer or after the current Start condition is sent.
In Slave mode:
0: No Stop generation.
1: Release the SCL and SDA lines after the current byte transfer.

Bit 8 **START**: Start generation
This bit is set and cleared by software and cleared by hardware when start is sent or PE=0.
In Master Mode:
0: No Start generation
1: Repeated start generation
In Slave mode:
0: No Start generation
1: Start generation when the bus is free

Bit 7 **NOSTRETCH**: Clock stretching disable (Slave mode)
This bit is used to disable clock stretching in slave mode when ADDR or BTF flag is set, until it is reset by software.
0: Clock stretching enabled
1: Clock stretching disabled

Bit 6 **ENGC**: General call enable
0: General call disabled. Address 00h is NACKed.
1: General call enabled. Address 00h is ACKed.

Bit 5 **ENPEC:** PEC enable
0: PEC calculation disabled
1: PEC calculation enabled

Bit 4 **ENARP**: ARP enable
0: ARP disable
1: ARP enable
SMBus Device default address recognized if SMBTYPE=0
SMBus Host address recognized if SMBTYPE=1

Bit 3 **SMBTYPE**: SMBus type
0: SMBus Device
1: SMBus Host

Bit 2 Reserved, must be kept at reset value

Bit 1 **SMBUS**: SMBus mode
0: I²C mode
1: SMBus mode

Bit 0 **PE**: Peripheral enable
0: Peripheral disable
1: Peripheral enable

Note: If this bit is reset while a communication is on going, the peripheral is disabled at the end of the current communication, when back to IDLE state.
All bit resets due to PE=0 occur at the end of the communication.
In master mode, this bit must not be reset before the end of the communication.

Note: When the STOP, START or PEC bit is set, the software must not perform any write access to I2C_CR1 before this bit is cleared by hardware. Otherwise there is a risk of setting a second STOP, START or PEC request.

27.6.2 I²C Control register 2 (I2C_CR2)

Address offset: 0x04
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Reserved		FREQ[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value

Bit 12 **LAST**: DMA last transfer
0: Next DMA EOT is not the last transfer
1: Next DMA EOT is the last transfer

Note: This bit is used in master receiver mode to permit the generation of a NACK on the last received data.

Bit 11 **DMAEN**: DMA requests enable
0: DMA requests disabled
1: DMA request enabled when TxE=1 or RxNE =1

Bit 10 **ITBUFEN**: Buffer interrupt enable
0: TxE = 1 or RxNE = 1 does not generate any interrupt.
1: TxE = 1 or RxNE = 1 generates Event Interrupt (whatever the state of DMAEN)

Bit 9 **ITEVTEN**: Event interrupt enable

0: Event interrupt disabled
1: Event interrupt enabled
This interrupt is generated when:

- SB = 1 (Master)
- ADDR = 1 (Master/Slave)
- ADD10= 1 (Master)
- STOPF = 1 (Slave)
- BTF = 1 with no TxE or RxNE event
- TxE event to 1 if ITBUFEN = 1
- RxNE event to 1if ITBUFEN = 1

Bit 8 **ITERREN**: Error interrupt enable

0: Error interrupt disabled
1: Error interrupt enabled
This interrupt is generated when:

- BERR = 1
- ARLO = 1
- AF = 1
- OVR = 1
- PECERR = 1
- TIMEOUT = 1
- SMBALERT = 1

Bits 7:6 Reserved, must be kept at reset value

Bits 5:0 **FREQ[5:0]**: Peripheral clock frequency

The FREQ bits must be configured with the APB clock frequency value (I2C peripheral connected to APB). The FREQ field is used by the peripheral to generate data setup and hold times compliant with the I2C specifications. The minimum allowed frequency is 2 MHz, the maximum frequency is limited by the maximum APB frequency and cannot exceed 50 MHz (peripheral intrinsic maximum limit).
0b000000: Not allowed
0b000001: Not allowed
0b000010: 2 MHz
...
0b110010: 50 MHz
Higher than 0b100100: Not allowed

27.6.3 I²C Own address register 1 (I2C_OAR1)

Address offset: 0x08
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD MODE	Reserved					ADD[9:8]		ADD[7:1]							ADD0
rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **ADDMODE** Addressing mode (slave mode)
0: 7-bit slave address (10-bit address not acknowledged)
1: 10-bit slave address (7-bit address not acknowledged)

Bit 14 Should always be kept at 1 by software.

Bits 13:10 Reserved, must be kept at reset value

Bits 9:8 **ADD[9:8]**: Interface address
7-bit addressing mode: don't care
10-bit addressing mode: bits9:8 of address

Bits 7:1 **ADD[7:1]**: Interface address
bits 7:1 of address

Bit 0 **ADD0**: Interface address
7-bit addressing mode: don't care
10-bit addressing mode: bit 0 of address

27.6.4 I²C Own address register 2 (I2C_OAR2)

Address offset: 0x0C
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ADD2[7:1]							ENDUAL
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value

Bits 7:1 **ADD2[7:1]**: Interface address
bits 7:1 of address in dual addressing mode

Bit 0 **ENDUAL**: Dual addressing mode enable
0: Only OAR1 is recognized in 7-bit addressing mode
1: Both OAR1 and OAR2 are recognized in 7-bit addressing mode

27.6.5 I²C Data register (I2C_DR)

Address offset: 0x10
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value

Bits 7:0 **DR[7:0]** 8-bit data register

Byte received or to be transmitted to the bus.

– Transmitter mode: Byte transmission starts automatically when a byte is written in the DR register. A continuous transmit stream can be maintained if the next data to be transmitted is put in DR once the transmission is started (TxE=1)

– Receiver mode: Received byte is copied into DR (RxNE=1). A continuous transmit stream can be maintained if DR is read before the next data byte is received (RxNE=1).

Note: In slave mode, the address is not copied into DR.

Write collision is not managed (DR can be written if TxE=0).

If an ARLO event occurs on ACK pulse, the received byte is not copied into DR and so cannot be read.

27.6.6 I²C Status register 1 (I2C_SR1)

Address offset: 0x14
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMB ALERT	TIME OUT	Res.	PEC ERR	OVR	AF	ARLO	BERR	TxE	RxNE	Res.	STOPF	ADD10	BTF	ADDR	SB
rc_w0	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	r	r		r	r	r	r	r

Bit 15 **SMBALERT**: SMBus alert

In SMBus host mode:
0: no SMBALERT
1: SMBALERT event occurred on pin
In SMBus slave mode:
0: no SMBALERT response address header
1: SMBALERT response address header to SMBALERT LOW received

– Cleared by software writing 0, or by hardware when PE=0.

Bit 14 **TIMEOUT**: Timeout or Tlow error

0: No timeout error
1: SCL remained LOW for 25 ms (Timeout)
or
Master cumulative clock low extend time more than 10 ms (Tlow:mext)
or
Slave cumulative clock low extend time more than 25 ms (Tlow:sext)

– When set in slave mode: slave resets the communication and lines are released by hardware
– When set in master mode: Stop condition sent by hardware
– Cleared by software writing 0, or by hardware when PE=0.

Note: This functionality is available only in SMBus mode.

Bit 13 Reserved, must be kept at reset value

Bit 12 **PECERR**: PEC Error in reception

0: no PEC error: receiver returns ACK after PEC reception (if ACK=1)
1: PEC error: receiver returns NACK after PEC reception (whatever ACK)

– Cleared by software writing 0, or by hardware when PE=0.

Note: When the received CRC is wrong, PECERR is not set in slave mode if the PEC control bit is not set before the end of the CRC reception. Nevertheless, reading the PEC value determines whether the received CRC is right or wrong.

Bit 11 **OVR**: Overrun/Underrun

0: No overrun/underrun
1: Overrun or underrun

– Set by hardware in slave mode when NOSTRETCH=1 and:
– In reception when a new byte is received (including ACK pulse) and the DR register has not been read yet. New received byte is lost.
– In transmission when a new byte should be sent and the DR register has not been written yet. The same byte is sent twice.
– Cleared by software writing 0, or by hardware when PE=0.

Note: If the DR write occurs very close to SCL rising edge, the sent data is unspecified and a hold timing error occurs

Bit 10 **AF**: Acknowledge failure

0: No acknowledge failure
1: Acknowledge failure

– Set by hardware when no acknowledge is returned.
– Cleared by software writing 0, or by hardware when PE=0.

Bit 9 **ARLO**: Arbitration lost (master mode)

0: No Arbitration Lost detected
1: Arbitration Lost detected

Set by hardware when the interface loses the arbitration of the bus to another master

– Cleared by software writing 0, or by hardware when PE=0.

After an ARLO event the interface switches back automatically to Slave mode (MSL=0).

Note: In SMBUS, the arbitration on the data in slave mode occurs only during the data phase, or the acknowledge transmission (not on the address acknowledge).

Bit 8 **BERR**: Bus error

0: No misplaced Start or Stop condition
1: Misplaced Start or Stop condition

– Set by hardware when the interface detects an SDA rising or falling edge while SCL is high, occurring in a non-valid position during a byte transfer.
– Cleared by software writing 0, or by hardware when PE=0.

Bit 7 **TxE**: Data register empty (transmitters)

0: Data register not empty
1: Data register empty

– Set when DR is empty in transmission. TxE is not set during address phase.
– Cleared by software writing to the DR register or by hardware after a start or a stop condition or when PE=0.

TxE is not set if either a NACK is received, or if next byte to be transmitted is PEC (PEC=1)

Note: TxE is not cleared by writing the first data being transmitted, or by writing data when BTF is set, as in both cases the data register is still empty.

Bit 6 **RxNE**: Data register not empty (receivers)

0: Data register empty
1: Data register not empty

– Set when data register is not empty in receiver mode. RxNE is not set during address phase.
– Cleared by software reading or writing the DR register or by hardware when PE=0.

RxNE is not set in case of ARLO event.

Note: RxNE is not cleared by reading data when BTF is set, as the data register is still full.

Bit 5 Reserved, must be kept at reset value

Bit 4 **STOPF**: Stop detection (slave mode)

0: No Stop condition detected
1: Stop condition detected

– Set by hardware when a Stop condition is detected on the bus by the slave after an acknowledge (if ACK=1).
– Cleared by software reading the SR1 register followed by a write in the CR1 register, or by hardware when PE=0

Note: The STOPF bit is not set after a NACK reception.
It is recommended to perform the complete clearing sequence (READ SR1 then WRITE CR1) after the STOPF is set. Refer to Figure 242.

Bit 3 **ADD10**: 10-bit header sent (Master mode)

0: No ADD10 event occurred.
1: Master has sent first address byte (header).

– Set by hardware when the master has sent the first byte in 10-bit address mode.
– Cleared by software reading the SR1 register followed by a write in the DR register of the second address byte, or by hardware when PE=0.

Note: ADD10 bit is not set after a NACK reception

Bit 2 **BTF**: Byte transfer finished

0: Data byte transfer not done
1: Data byte transfer succeeded

– Set by hardware when NOSTRETCH=0 and:
– In reception when a new byte is received (including ACK pulse) and DR has not been read yet (RxNE=1).
– In transmission when a new byte should be sent and DR has not been written yet (TxE=1).
– Cleared by software by either a read or write in the DR register or by hardware after a start or a stop condition in transmission or when PE=0.

Note: The BTF bit is not set after a NACK reception

The BTF bit is not set if next byte to be transmitted is the PEC (TRA=1 in I2C_SR2 register and PEC=1 in I2C_CR1 register)

Bit 1 **ADDR**: Address sent (master mode)/matched (slave mode)

This bit is cleared by software reading SR1 register followed reading SR2, or by hardware when PE=0.

Address matched (Slave)

0: Address mismatched or not received.
1: Received address matched.

– Set by hardware as soon as the received slave address matched with the OAR registers content or a general call or a SMBus Device Default Address or SMBus Host or SMBus Alert is recognized. (when enabled depending on configuration).

Note: In slave mode, it is recommended to perform the complete clearing sequence (READ SR1 then READ SR2) after ADDR is set. Refer to Figure 242.

Address sent (Master)

0: No end of address transmission
1: End of address transmission

– For 10-bit addressing, the bit is set after the ACK of the 2nd byte.
– For 7-bit addressing, the bit is set after the ACK of the byte.

Note: ADDR is not set after a NACK reception

Bit 0 **SB**: Start bit (Master mode)

0: No Start condition
1: Start condition generated.

– Set when a Start condition generated.
– Cleared by software by reading the SR1 register followed by writing the DR register, or by hardware when PE=0

27.6.7 I²C Status register 2 (I2C_SR2)

Address offset: 0x18
Reset value: 0x0000

Note: Reading I2C_SR2 after reading I2C_SR1 clears the ADDR flag, even if the ADDR flag was set after reading I2C_SR1. Consequently, I2C_SR2 must be read only when ADDR is found set in I2C_SR1 or when the STOPF bit is cleared.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEC[7:0]								DUALF	SMB HOST	SMBDE FAULT	GEN CALL	Res.	TRA	BUSY	MSL
r	r	r	r	r	r	r	r	r	r	r	r		r	r	r

Bits 15:8 **PEC[7:0]** Packet error checking register
This register contains the internal PEC when ENPEC=1.

Bit 7 **DUALF**: Dual flag (Slave mode)
0: Received address matched with OAR1
1: Received address matched with OAR2
– Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 6 **SMBHOST**: SMBus host header (Slave mode)
0: No SMBus Host address
1: SMBus Host address received when SMBTYPE=1 and ENARP=1.
– Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 5 **SMBDEFAULT**: SMBus device default address (Slave mode)
0: No SMBus Device Default address
1: SMBus Device Default address received when ENARP=1
– Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 4 **GENCALL**: General call address (Slave mode)
0: No General Call
1: General Call Address received when ENGC=1
– Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 3 Reserved, must be kept at reset value

Bit 2 **TRA**: Transmitter/receiver
0: Data bytes received
1: Data bytes transmitted
This bit is set depending on the R/W bit of the address byte, at the end of total address phase.
It is also cleared by hardware after detection of Stop condition (STOPF=1), repeated Start condition, loss of bus arbitration (ARLO=1), or when PE=0.

Bit 1 **BUSY**: Bus busy
0: No communication on the bus
1: Communication ongoing on the bus
– Set by hardware on detection of SDA or SCL low
– cleared by hardware on detection of a Stop condition.
It indicates a communication in progress on the bus. This information is still updated when the interface is disabled (PE=0).

Bit 0 **MSL**: Master/slave
0: Slave Mode
1: Master Mode
– Set by hardware as soon as the interface is in Master mode (SB=1).
– Cleared by hardware after detecting a Stop condition on the bus or a loss of arbitration (ARLO=1), or by hardware when PE=0.

Note: Reading I2C_SR2 after reading I2C_SR1 clears the ADDR flag, even if the ADDR flag was set after reading I2C_SR1. Consequently, I2C_SR2 must be read only when ADDR is found set in I2C_SR1 or when the STOPF bit is cleared.

27.6.8 I²C Clock control register (I2C_CCR)

Address offset: 0x1C
Reset value: 0x0000

Note: *f_{PCLK1} must be at least 2 MHz to achieve Sm mode I²C frequencies. It must be at least 4 MHz to achieve Fm mode I²C frequencies. It must be a multiple of 10MHz to reach the 400 kHz maximum I²C Fm mode clock.*

The CCR register must be configured only when the I2C is disabled (PE = 0).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	Reserved		CCR[11:0]											
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **F/S:** I2C master mode selection
0: Sm mode I2C
1: Fm mode I2C

Bit 14 **DUTY:** Fm mode duty cycle
0: Fm mode t_{low}/t_{high} = 2
1: Fm mode t_{low}/t_{high} = 16/9 (see CCR)

Bits 13:12 Reserved, must be kept at reset value

Bits 11:0 **CCR[11:0]:** Clock control register in Fm/Sm mode (Master mode)
Controls the SCL clock in master mode.
Sm mode or SMBus:
$T_{high} = CCR * T_{PCLK1}$
$T_{low} = CCR * T_{PCLK1}$
Fm mode:
If DUTY = 0:
$T_{high} = CCR * T_{PCLK1}$
$T_{low} = 2 * CCR * T_{PCLK1}$
If DUTY = 1: (to reach 400 kHz)
$T_{high} = 9 * CCR * T_{PCLK1}$
$T_{low} = 16 * CCR * T_{PCLK1}$
For instance: in Sm mode, to generate a 100 kHz SCL frequency:
If FREQR = 08, T_{PCLK1} = 125 ns so CCR must be programmed with 0x28
(0x28 <=> 40d x 125 ns = 5000 ns.)

Note: *The minimum allowed value is 0x04, except in FAST DUTY mode where the minimum allowed value is 0x01*

$t_{high} = t_{r(SCL)} + t_{w(SCLH)}$. See device datasheet for the definitions of parameters.
$t_{low} = t_{f(SCL)} + t_{w(SCLL)}$. See device datasheet for the definitions of parameters.
I2C communication speed, fSCL ~ 1/(thigh + tlow). The real frequency may differ due to the analog noise filter input delay.
The CCR register must be configured only when the I²C is disabled (PE = 0).

27.6.9 I²C TRISE register (I2C_TRISE)

Address offset: 0x20
Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										TRISE[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 15:6 Reserved, must be kept at reset value

Bits 5:0 **TRISE[5:0]**: Maximum rise time in Fm/Sm mode (Master mode)

These bits should provide the maximum duration of the SCL feedback loop in master mode. The purpose is to keep a stable SCL frequency whatever the SCL rising edge duration.
These bits must be programmed with the maximum SCL rise time given in the I²C bus specification, incremented by 1.
For instance: in Sm mode, the maximum allowed SCL rise time is 1000 ns.
If, in the I2C_CR2 register, the value of FREQ[5:0] bits is equal to 0x08 and T_{PCLK1} = 125 ns therefore the TRISE[5:0] bits must be programmed with 09h.
(1000 ns / 125 ns = 8 + 1)
The filter value can also be added to TRISE[5:0].
If the result is not an integer, TRISE[5:0] must be programmed with the integer part, in order to respect the t_{HIGH} parameter.

Note: TRISE[5:0] must be configured only when the I2C is disabled (PE = 0).

27.6.10 I²C FLTR register (I2C_FLTR)

Address offset: 0x24

Reset value: 0x0000

The I2C_FLTR is available on STM32F42xxx and STM32F43xxx only.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											ANOFF	DNF[3:0]			
											rw	rw	rw	rw	rw

Bits 15:5 Reserved, must be kept at reset value

Bit 4 **ANOFF**: Analog noise filter OFF

0: Analog noise filter enable
1: Analog noise filter disable

Note: ANOFF must be configured only when the I2C is disabled (PE = 0).

Bits 3:0 **DNF[3:0]**: Digital noise filter

These bits are used to configure the digital noise filter on SDA and SCL inputs. The digital filter will suppress the spikes with a length of up to DNF[3:0] * TPCLK1.

0000: Digital noise filter disable
0001: Digital noise filter enabled and filtering capability up to 1* TPCLK1.
...
1111: Digital noise filter enabled and filtering capability up to 15* TPCLK1.

Note: DNF[3:0] must be configured only when the I2C is disabled (PE = 0). If the analog filter is also enabled, the digital filter is added to the analog filter.

27.6.11 I²C register map

The table below provides the I²C register map and reset values.

Table 125. I²C register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**I2C_CR1**	Reserved																SWRST	Reserved	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	Reserved	SMBUS	PE
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0		0	0
0x04	**I2C_CR2**	Reserved																			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Reserved	FREQ[5:0]						
	Reset value																				0	0	0	0	0		0	0	0	0	0	0	0
0x08	**I2C_OAR1**	Reserved																ADDMODE	Reserved					ADD[9:8]		ADD[7:1]							ADD0
	Reset value																	0						0	0	0	0	0	0	0	0	0	0
0x0C	**I2C_OAR2**	Reserved																								ADD2[7:1]							ENDUAL
	Reset value																									0	0	0	0	0	0	0	0
0x10	**I2C_DR**	Reserved																								DR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	**I2C_SR1**	Reserved																SMBALERT	TIMEOUT	Reserved	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Reserved	STOPF	ADD10	BTF	ADDR	SB
	Reset value																	0	0		0	0	0	0	0	0	0		0	0	0	0	0
0x18	**I2C_SR2**	Reserved																PEC[7:0]								DUALF	SMBHOST	SMBDEFAUL	GENCALL	Reserved	TRA	BUSY	MSL
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x1C	**I2C_CCR**	Reserved																F/S	DUTY	Reserved	CCR[11:0]												
	Reset value																	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**I2C_TRISE**	Reserved																												TRISE[5:0]			
	Reset value																												0	0	0	1	0
0x24	**I2C_FLTR**	Reserved																											ANOFF	DNF[3:0]			
	Reset value																												0	0	0	0	0

Refer to *Section 2.3: Memory map* for the register boundary addresses table.

28 Serial peripheral interface (SPI)

This section applies to the whole STM32F4xx family, unless otherwise specified.

28.1 SPI introduction

The SPI interface provides two main functions, supporting either the SPI protocol or the I^2S audio protocol. By default, it is the SPI function that is selected. It is possible to switch the interface from SPI to I^2S by software.

The serial peripheral interface (SPI) allows half/ full-duplex, synchronous, serial communication with external devices. The interface can be configured as the master and in this case it provides the communication clock (SCK) to the external slave device. The interface is also capable of operating in multimaster configuration.

It may be used for a variety of purposes, including simplex synchronous transfers on two lines with a possible bidirectional data line or reliable communication using CRC checking.

The I^2S is also a synchronous serial communication interface. It can address four different audio standards including the I^2S Philips standard, the MSB- and LSB-justified standards, and the PCM standard. It can operate as a slave or a master device in full-duplex mode (using 4 pins) or in half-duplex mode (using 3 pins). Master clock can be provided by the interface to an external slave component when the I^2S is configured as the communication master.

Warning: **Since some SPI1 and SPI3/I2S3 pins may be mapped onto some pins used by the JTAG interface (SPI1_NSS onto JTDI, SPI3_NSS/I2S3_WS onto JTDI and SPI3_SCK/I2S3_CK onto JTDO), you may either:**
– map SPI/I2S onto other pins
– disable the JTAG and use the SWD interface prior to configuring the pins listed as SPI I/Os (when debugging the application) or
– disable both JTAG/SWD interfaces (for standalone applications).
For more information on the configuration of the JTAG/SWD interface pins, please refer to *Section 8.3.2: I/O pin multiplexer and mapping*.

28.2 SPI and I²S main features

28.2.1 SPI features

- Full-duplex synchronous transfers on three lines
- Simplex synchronous transfers on two lines with or without a bidirectional data line
- 8- or 16-bit transfer frame format selection
- Master or slave operation
- Multimaster mode capability
- 8 master mode baud rate prescalers (f_{PCLK}/2 max.)
- Slave mode frequency (f_{PCLK}/2 max)
- Faster communication for both master and slave
- NSS management by hardware or software for both master and slave: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- SPI TI mode
- Hardware CRC feature for reliable communication:
  - CRC value can be transmitted as last byte in Tx mode
  - Automatic CRC error checking for last received byte
- Master mode fault, overrun and CRC error flags with interrupt capability
- 1-byte transmission and reception buffer with DMA capability: Tx and Rx requests

28.2.2 I²S features

- Full duplex communication
- Half-duplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler to reach accurate audio sample frequencies (from 8 kHz to 192 kHz)
- Data format may be 16-bit, 24-bit or 32-bit
- Packet frame is fixed to 16-bit (16-bit data frame) or 32-bit (16-bit, 24-bit, 32-bit data frame) by audio channel
- Programmable clock polarity (steady state)
- Underrun flag in slave transmission mode, overrun flag in reception mode (master and slave), and Frame Error flag in reception and transmission mode (slave only)
- 16-bit register for transmission and reception with one data register for both channel sides
- Supported I²S protocols:
  - I²S Phillps standard
  - MSB-justified standard (left-justified)
  - LSB-justified standard (right-justified)
  - PCM standard (with short and long frame synchronization on 16-bit channel frame or 16-bit data frame extended to 32-bit channel frame)
- Data direction is always MSB first
- DMA capability for transmission and reception (16-bit wide)
- Master clock may be output to drive an external audio component. Ratio is fixed at 256 × F_S (where F_S is the audio sampling frequency)
- Both I²S (I2S2 and I2S3) have a dedicated PLL (PLLI2S) to generate an even more accurate clock.
- I²S (I2S2 and I2S3) clock can be derived from an external clock mapped on the I2S_CKIN pin.